توضيح

كويك بيسك

تأليف

هاري وموراث تانيك واودو بوش

ترجمة ومراجعة

د. م. سرور علي ابراهيم سرور
الاستاذ المشارك بقسم الأساليب الكمية
كلية الاقتصاد والادارة
جامعة الملك سعود - فرع القصيم

تقديم

د. عبد الله بن عبد الله العبيد
عميد كلية الاقتصاد والادارة
جامعة الملك سعود - فرع القصيم



الناشر
المكتبة الاكاديمية
١٩٩٤

لوضي

كويك بيسك

بسم الله الرحمن الرحيم

تقديم

الحمد لله رب العالمين، والصلاة والسلام على أشرف المرسلين سيدنا محمد وعلى آله وصحبه ومن تبعهم بإحسان إلى يوم الدين.

وبعد...

فهذا كتاب جديد عن إحدى لغات الكمبيوتر التى بدأ استخدامها منذ خمسة وعشرين عاماً ولكن شتان بين الصيغة الأولى لهذه اللغة والصيغة المعروضة حالياً فى هذا الكتاب. فقد ظهرت لغة البيسك كلغة مبسطة لتعليم المبتدئين كتابة برامج للكمبيوتر. وقد كانت امكانيات هذه اللغة مبسطة لدرجة جعلت من غير المتوقع أن تأخذ هذه اللغة شكلها الحالى. وكما هو الشأن فى مجال الكمبيوتر دائماً فلم تأخذ لغة البيسك شكلها الحالى فجأة بل حدث هذا من خلال التطور المستمر لهذه اللغة والذى جعل البعض فى بداية الثمانينيات الميلادية يعتقد أنه من المستحيل إعداد صيغة نمطية لهذه اللغة لا لشىء إلا لكثرة الصيغ التجارية المتوفرة لها فى الأسواق. إلا أن البعض اقترح أن تكون صيغة بيسك أجهزة الميكروكمبيوتر من طراز IBM (IBM PC BASIC) وصيغة البيسك المسماة GW-BASIC من شركة ميكروسوفت كصيغ نمطية للغة البيسك. إلا أن صيغ بيسك السريع المقدمة من شركة ميكروسوفت وكذلك صيغ تربو بيسك المقدمة من شركة بورلاند العالمية لم تغير فقط من محتويات لغة البيسك بل أنها ادخلتها كذلك فى مصاف اللغات المهمة المتوقع الاعتماد عليها بشدة فى المستقبل. وأحد الأسباب الرئيسية لذلك هو أن هذه الصيغ من لغة البيسك هى صيغ مرتبة structured تصلح للبرمجة المرتبة.

ولم يقع اختيار الكلية على هذا الكتاب لترجمته وتقديمه للقارئ العربى لحداثة المادة العلمية المقدمة فيه فحسب بل لسهولة عرضها وترتيب مواضيعها. وعلى الرغم من أن الكتاب مقدم للقارئ المبتدئ وكذلك للمبرمجين إلا أنه لا يحتاج لشئ فى دراسته سوى الالمام باستخدام لوحة مفاتيح الكمبيوتر ونظام تشغيله. وهذه متطلبات ميسورة لمن يرغب فى تعلم هذه اللغة المهمة.

ولا يفوتنا فى هذه السطور أن ننوه بالجهد العلمى المشكور الذى قام به الدكتور/ سرور على ابراهيم سرور الاستاذ المشارك بقسم الأساليب الكمية بالكلية، حيث أخذ على عاتقه مسئولية

الكتاب كاملة فى الترجمة ومراجعة الترجمة أيضاً، فجزاه الله خير الجزاء على هذا العمل العلمى فى ميدان من أهم ميادين العلم فى حياتنا المعاصرة.

نسأل الله تعالى أن يجعل كل أعمالنا خاصة لوجهه، إنه سميع مجيب.

د. عبد الله بن عبد الله العبيد
عميد كلية الاقتصاد والادارة
جامعة الملك سعود – فرع القصيم

بسم الله الرحمن الرحيم

« قل لو كان البحر مداداً لكلمات ربى لنفد البحر
قبل أن تنفد كلمات ربى ولو جئنا بمثله مدداً ».

صدق الله العظيم

مقدمة المترجم

بسم الله الرحمن الرحيم

الحمد لله، نحمده ونستعينه، ونصلى ونسلم على خير انبيائه، وخاتم رسله.

وبعد ....

لقد ظهرت لغة البيسك منذ خمسة وعشرين عاماً كلغة بسيطة لتعليم المبتدئين أساسيات البرمجة باستخدام الكمبيوتر. وشهدت هذه اللغة البسيطة تطورات هائلة على مدار هذه السنين جعلتها فى مصاف لغات البرمجة الأساسية حالياً والمتوقع الإكثار من الاعتماد عليها مستقبلياً بعد أن أصبحت تدعم البرمجة المرتبة والتى كانت تفتقر إليها الصيغ السابقة لهذه اللغة. ويرجع الفضل فى جعل هذه اللغة قادرة على دعم البرمجة المرتبة إلى صيغ كويك بيسك أو بيسك السريع المقدمة من شركة ميكروسوفت وإلى صيغ تربو بيسك المقدمة من شركة بورلاند العالمية. ويقدم هذا الكتاب، على هيئة دروس عملية مبسطة، الصيغة رقم 4.0 من لغة بيسك السريع، وسوف يستخدم اسم بيسك السريع فى بقية الكتاب.

وفيما يلى بعض مميزات صيغ بيسك السريع عن صيغ بيسك السابقة لها وبصفة خاصة صيغ IBM PC BASIC و GW-BASIC والتى يعتبرها البعض صيغاً نمطية للغة البيسك :

١ - إمكانية كتابة الأسطر دون وضع أرقام لها. وامكانية استخدام الفاصلة أو REM فى كتابة الملاحظات. واستخدام اشباه الأوامر، واستخدام منظومات استاتيكية ومنظومات ديناميكية.

٢ - استخدام نوع جديد من أنواع المتغيرات العددية وهو الرقم الصحيح الطويل long integer وكذلك استخدام سلاسل وسجلات ثابتة الطول، والعمل من خلال قوائم وامكانية الحصول على مساعدة أثناء العمل.

٣ - التوسع فى مكونات التحكم التى سبق استخدامها فى صيغ البيسك السابقة لها بادخال مجموعات IF ومجموعات SELECT CASE ودورات DO.

٤ - استخدام البرامج الفرعية والتى تدعم البرمجة المرتبة، وتكمن قوة البرامج الفرعية فى امكانية تمرير قيم إلى البرامج الفرعية مع بقاء متغيرات البرنامج الفرعى محلية له، كما ينطبق نفس المفهوم على الدوال كذلك.

٥ - استخدام الملفات الثنائية، واستخدام مترجمات compilers للغة لترجمة البرامج ويمكن تنفيذ هذه البرامج اما من بيئة اللغة واما من بيئة نظام التشغيل DOS مباشرة.

٦ - استخدام المقاطع التى يمكن معالجتها كأجزاء مستقلة، واستخدام خاصية مشاركة المتغيرات عبر المقاطع المختلفة، وتجميع المقاطع المترجمة فى مكتبة خاصة بها للاستعانة بها عند الحاجة لذلك، وامكانية توصيل المقاطع المنفصلة مع بعضها البعض أو مع مقاطع المكتبة أو مع برامج بيسك سريع أخرى.

هذا وقد راعينا فى ترجمة الكتاب الحفاظ على نفس الأسلوب والتنظيم المتبعين من قبل المؤلف. كما حافظنا كذلك على نفس التسلسل الذى قدمه المؤلف حيث تم سرد دروس الكتاب فى تسلسلها طبقاً للحروف الأبجدية الانجليزية، كما تمت ترجمة تسلسل التعلم الذى يقترحه مؤلف الكتاب لدراسة الكتاب كدليل لمساعدة القارئ العربى المبتدئ فى التعامل مع الكتاب.

ولا يسعنا فى نهاية هذه المقدمة إلا أن نقدم خالص الشكر والعرفان إلى كل من ساهم فى إخراج هذا الكتاب فى هذه الصورة ونخص بالذكر سعادة الدكتور/ عبد الله بن عبد الله العبيد عميد كلية الاقتصاد والإدارة بجامعة الملك سعود فرع القصيم بعنيزة لتشجيعه الدائم على ترجمة أمهات الكتب إلى اللغة العربية وحثه المستمر لأعضاء هيئة التدريس بالكلية على الاستمرار فى نفس هذا النهج. كما نشكر الاستاذ/ محمد رضوان لما ساهم به فى انجاز هذا الكتاب. ولايفوتنا أن نشكر سعادة الاستاذ/ أحمد أمين محمود- مدير المكتبة الاكاديمية بالقاهرة- وسعادة المهندس/ حمدى قنديل، مدير الانتاج بالمكتبة الاكاديمية بالقاهرة، وجميع العاملين بالمكتبة لما يبذلونه من جهد صادق فى اخراج مثل هذه الكتب العلمية القيمة للمكتبة العربية. ونرجو من الله أن نكون قد وفقنا لتقديم كتاب جيد وجديد للقارئ العربى.

والله ولى التوفيق.

المترجم

المحتويات

تسلسل التعلم المقترح اتباعه

التسلسل	الدرس	رقم الصفحة
١	الأول	٢٥
٢	الثاني	٢٧
٣	ملحق B	٥٠٦
٤	الثالث	٣١
٥	الرابع	٣٧
٦	الثالث والعشرون	٩٣
٧	المائة والخامس والخمسون	٤٧٧
٨	المائة والثالث والخمسون	٤٧٢
٩	الخامس والسبعون	٢٦٢
١٠	السادس والخمسون	٢٠٥
١١	السبعون	٢٤٨
١٢	التاسع والسبعون	٢٧٣
١٣	المائة وتسعة	٣٥٣
١٤	الخامس عشر	٧٠
١٥	المائة والثالث والأربعون	٤٤٥
١٦	المائة والثالث والثلاثون	٤٢٤
١٧	التاسع والأربعون	١٧٩
١٨	الثامن والخمسون	٢١٠
١٩	المائة والسادس والعشرون	٤٠٦
٢٠	التاسع عشر	٧٩
٢١	الخامس والخمسون	٢١٠
٢٢	السادس والثلاثون	١٣٨
٢٣	المائة والسابع والأربعون	٤٥٧
٢٤	المائة والرابع والثلاثون	٤٢٦
٢٥	السادس	٤٢

الدرس الأول

حول هذا الكتاب

مقدمة

يصف هذا الكتاب مترجم البيسك للصيغة الرابعة version 4.0 من البيسك السريع Quick BASIC الذى أعدته شركة ميكروسوفت Microsoft وبيئة إعداد البرامج بهذه اللغة. ويوجد لمترجم بيسك السريع سطح بينى بسيط للمستفيد وكذلك تكوين بسيط للقوائم. ويخدم تصميمه فى مساعدة المستفيد فى إعداد برامجه بأقل عدد من الخطوات. ويتأكد المنقح الذكى من التكوين مع ادخال أسطر البرنامج. ويترجم البرنامج طبقاً لما كتب عليه مع السماح بتشغيل البرنامج فى خطوة واحدة. وتعرف الأخطاء مع سردها فى تقرير بطريقة تسمح للمستفيد أن يصححها على الفور.

ويوضح بيسك السريع المشروح هنا استخدام الدوال والإجراءات المختلفة المتاحة كجزء من مترجم بيسك السريع. وقد صمم الكتاب ليستخدمه المبتدئون والمهنيون كذلك. كما أنه صمم ليخدم كوسيلة للأساتذة فى قاعات التدريس. ويفترض هذا الكتاب أن هناك معرفة بالعمل باستخدام لوحة مفاتيح الكمبيوتر ونظام التشغيل DOS من قبل القارىء.

التنظيم

لخدمة القاعدة العريضة من المستفيدين المختلفين المقدم لهم هو الكتاب فقد تم تنظيمه فى دروس صغيرة سهلة القراءة. وأنت تقرأ الدرس الأول الآن.

والدرس الثانى عبارة عن عرض عام للصيغة 4.0 من بيسك السريع. ويقدم هذا الدرس معلومات عن محتويات الأقراص واقتراحات لإعداد الكمبيوتر المتاح لك لاستخدام بيسك السريع.

الدرس الثالث، وهو عينة لجلسة مع بيسك السريع، يضع يدك على التوضيح. وتستطيع من هناك ان تشعر بتكامل بيسك السريع ومنقح الشاشة الكاملة وقائمة التشغيل Run. وتتعلم كيفية عمل ملف تنفيذى EXE. من برامجك بحيث يمكنك أن تقوم بتشغيل مثل هذه البرامج بدون استخدام بيسك السريع. ولاكتساب الخبرة بمدى بساطة امكانية استخدام بيسك السريع فقد ترغب فى القفز إلى الدرس الثالث مباشرة لرؤية ذلك بنفسك.

وتناقش الدروس المتبقية الدوال والاجراءات وتكوينات اللغة المختلفة المتاحة للمستفيد من بيسك السريع. فهى تقدم أوصافاً وتطبيقات وعمليات نمطية للبرامج للمساعدة فى حل مشاكل عملية يومية. كما تقدم عديداً من الأمثلة المفيدة والممتعة فى هذا الكتاب والتى يمكن أن تكون وسائل مرتفعة القيمة عندما تقوم ببناء برامج أكبر وأكثر تعقيداً. وقد تم اختبار كل الأمثلة ووجد أنها تعمل كلها بطريقة صحيحة. بالنسبة للمبتدئين فهذا يساعد فى التغلب على الصدمة الأولى لمحاولة استخدام دالة جديدة أو تركيبة جديدة بدون نجاح. والدروس مرتبة ترتيباً أبجدياً (طبقاً للحروف الانجليزية) لسهولة الرجوع إليها والاتصال بأوامر محددة وأمثلة برمجة معينة.

وفى بداية الكتاب يوجد تسلسل للتعلم يوصى باتباعه. فهذا التسلسل يرتب الدروس ترتيباً منطقياً للمبتدئين وذلك ليتعودوا على استخدام بيسك السريع خطوة بخطوة أو ليستخدمه الاساتذة فى قاعات الدراسة كخطوط عريضة للمقرر.

وتقدم الملاحق من A إلى G معلومات اضافية عن بيسك السريع وتمارين برمجة لكل الدروس.

ملحق A عبارة عن قائمة بالمصطلحات المستخدمة فى الكتاب ووصف موجز لها. وملحق B عبارة عن مقدمة لبيئة تطوير بيسك السريع. وينصح القارىء بقراءة هذا الملحق أكثر من مرة واحدة. ويحتوى ملحق C على جدول وقائمة بكلمات بيسك السريع الرئيسية. ويناقش ملحق D مترجم أسطر الأوامر والواصل وخيارات الترجمة المختلفة المتاحة. ويناقش ملحق E انتاج وصيانة مكتبات بيسك السريع. ويسرد ملحق F رسائل الخطأ التى تنتج بواسطة بيسك السريع. كما يقدم ملحق G تمارين مصممة لاختبار المعلومات التى تم تعلمها من هذه الدروس.

متطلبات نظم المكونات ونظم البرامج

يحتاج مترجم بيسك السريع من شركة ميكروسوفت إلى ما يلى :

- جهاز كمبيوتر IBM PC أو أى جهاز آخر متوافق معه تماماً ويستخدم نظام تشغيل MS-DOS أو PC-DOS الصيغ من 2.0 فاعلى منها.
- مشغل أقراص مرنة واحد على الأقل ويفضل وجود مشغلين.
- ذاكرة اتصال عشوائى (RAM) لا تقل عن 320 كيلوبايت.

انتقل إلى الدرس الثانى للاستمرار فى تسلسل التعلم الموصى به.

الدرس الثانى

عرض عام لبيسك السريع

مقدمة

لقد كانت شركة ميكروسوفت رائدة فى تطوير نظم البرامج المبتكرة لأجهزة الميكروكمبيوتر. وفى تقليدها هذا فإن بيسك السريع (الصيغة 4.0) هو منتج يمثل نقطة تحول ويقدم بيئة إعداد برامج متكاملة ومذهلة بلغة البيسك.

وتشمل البيئة شاشة كاملة ومنقحاً للتأكد من التكوين وامكانيات التنقيح لملفات عديدة ونوافذ عديدة وتسهيلات للتصحيح الكامل وقوائم للسحب لأسفل وتكوين قائمة سهل وقوى يمكن قيادته من خلال لوحة المفاتيح أو باستخدام فأرة. وبمجرد أن تتواجد فى بيئة بيسك السريع فيمكنك أن تنقح البرنامج وتقوم بتشغيله وتصحيحه بدون أن تغير من البرنامج أو الاقراص.

وبيسك المفسر يكون متسعاً فى هذا التفسير للبيسك فى عديد من الأوجه :

- تسمح لك البرامج الفرعية بتجزئة البرامج إلى أجزاء فردية ومستقلة منطقياً.
- دعم البرمجة المرتبة مقدم فى صورة IF.. THEN متعددة الأسطر ومكونات تحكم مساراً مثل WHILE.. WEND و SUB واجراءات FUNCTION وعبارات TYPE.. END TYPE لمتغيرات يعرفها المستفيد واجراءات اعادة ذاتية.
- يمكن استخدام عناوين حرفية عددية تجعل البرنامج أكثر سهولة فى قراءته وكتابته لأن بيسك السريع لا يتطلب أرقاماً للأسطر الخاصة بعبارات برامجه.
- يمكن اقرار المنظومات الديناميكية لاستغلال الذاكرة استغلالاً أمثل.
- يسمح بمنظومات عديدة كبيرة. فيمكن استخدام أى عدد من منظومات حجمها 64K والتى تتسق مع الذاكرة المتاحة.

يسمح بأن تقوم ما تسمى باشباه الأوامر metacommands بالتحكم فى الطريقة التى يفسر ويترجم بها المترجم برنامجك. مثال ذلك، يجعلك الأمر $INCLUDE تدخل ملفات مصدر اضافية أثناء الترجمة.

محتويات الأقراص :

يأتي بيسك السريع (الصيغة 4.0) في ثلاثة أقراص مسماة بقرص البرنامج وقرص المكتبات وقرص المنافع والأمثلة على التوالي. وفيما يلي قائمة بمحتويات كل قرص :

القرص الأول : قرص البرنامج

Disk 1

Filename	*Description*
BC.EXE	(BASIC command-line compiler)
QB.EXE	(QuickBASIC program)
SETUP.BAT	(QuickBASIC installation program)
SETUP1.BAT	(" " ")
PACKING.LST	(List of files on the disks)
README.DOC	(Latest information on QuickBASIC)

القرص الثاني : قرص المكتبات

Disk 2

Filename	*Description*
BRUN40.EXE	(QuickBASIC run-time module)
BRUN40.LIB	(QuickBASIC-library run-time support)
BCOM40.LIB	(QuickBASIC alternate run-time library)
BQLB40.LIB	(Quick-library run-time support)

القرص الثالث : قرص المنافع والأمثلة

Disk 3

Filename	*Description*
SOURCE	< DIR>
LIB.EXE	(Microsoft library manager)
LINK.EXE	(Microsoft overlay linker)
QB.HLP	(Help text for QuickBASIC)
MOUSE.COM	(Mouse driver)
NOEM.OBJ	(Programming support file)
REMLINE.BAS	(Example program)
SORTDEMO.BAS	(" ")
TORUS.BAS	(" ")
ABSOLUTE.ASM	(" ")
INTRPT.ASM	(" ")
QB.LIB	(QuickBASIC library; supports DOS interrupts)
QB.QLB	(Quick-library; supports DOS interrupts)
QB.PIF	(QuickBASIC system file)
QB.BI	(" " ")
NOCOM.OBJ	(Programming support file)
DEMO1.BAS	(" ")
DEMO2.BAS	(" ")
DEMO3.BAS	(" ")
QBHERC.COM	
FIXSHIFT.COM	(" ")

Contents of Directory Source

Filename	*Description*
INDEX.BAS	(Example program)
CAL.BAS	(" ")
CUBE.BAS	(" ")
EDPAT.BAS	(" ")
MANDEL.BAS	(" ")
SINEWAVE.BAS	(" ")
BAR.BAS	(" ")
PALETTE.BAS	(" ")
BALLPSET.BAS	(" ")
WHEREIS.BAS	(" ")
CRLF.BAS	(" ")
BALLXOR.BAS	(" ")
COLORS.BAS	(" ")
FILERR.BAS	(" ")
PLOTTER.BAS	(" ")
STRTONUM.BAS	(" ")
TERMINAL.BAS	(" ")
CHECK.BAS	(" ")
QLBDUMP.BAS	(" ")
SEARCH.BAS	(" ")
FLPT.BAS	(" ")
TOKEN.BAS	(" ")
ENTAB.BAS	(" ")
HIDE.BAS	(" ")

وقد تحتوى أقراصك على أمثلة أخرى وبرامج أخرى للأغراض العامة ويكون لكل برامج البيسك التوسع BAS. اقرأ الملف PACKING.LIST لمعرفة المزيد عن محتويات الأقراص.

الإعداد الموصى به

اعتماداً على تشكيل نظام الكمبيوتر المتاح لك يمكن إعداد بيسك السريع بطرق عديدة. وفيما يلى طرق اعداد يوصى بها للثلاثة تشكيلات الممكنة :

نظام به مشغلين اقراص مرنة : لاستخدام بيسك السريع مع نظام قرصين مرنين اتبع الخطوات التالية :

* استخدم المشغل B فى تخزين ملفات المصدر بالبيسك.
* استخدم المشغل A لبرامج بيسك السريع.
* قم بتشغيل بيسك السريع من المشغل B.

يضمن لك هذا أن بيسك السريع يستطيع أن يجد أى مكتبات قد يحتاج إليها برنامجك وتكتب ملفات المصدر والتشغيل فى المشغل B.

– نظام مشغل أقراص مرنة واحد : هناك طريقتان :

الطريقة الأولى : خزن الملفات اللازمة لتشغيل بيسك السريع (مثل QB.EXE و QB.HLP و COMMAND.COM) وملفات المصدر بالبيسك على قرص واحد.

الطريقة الثانية : لاستخدام بيسك السريع قم بتشغيل بيسك السريع من قرص البرنامج. وبعد تحميل بيسك السريع استبدل قرص البرنامج بقرص فارغ واستخدم هذا القرص فى تخزين برامجك المكتوبة بالبيسك.

– نظام قرص صلب : حيث إن تنظيم القرص الصلب يعتمد داخلياً على الاستخدام الفردى له فما يلى هو طريقة واحدة فقط لاعداد الأدلة لاستخدام بيسك السريع :

* قم بإعداد دليل directory (QB\) من دليل الجذر root directory.
* استخدم هذا الدليل لكل ملفات برامج بيسك السريع وبرامجك.

اجعل المسار DOS PATH يحتوى على دليل بيسك السريع C: \QB فى مسار بحثه. مثال ذلك :

```
Set Path=C:\;C:\QB;
```

والبديل لذلك هو أنه يمكنك استخدام ملف دفعة للأعداد (SETUP1.BAT, SETUP.BAT) لوضع بيسك السريع على القرص الصلب.

انتقل إلى الملحق B للاستمرار فى تسلسل التعلم.

الدرس الثالث

عينة لجلسة مع بيسك السريع

مقدمة

تقدمك عينة الجلسة هذه لمترجم بيسك السريع من شركة ميكروسوفت وتعودك هذه الجلسة على السطح البينى للمستفيد وتكوين قائمة البيئة المتكاملة للبيسك السريع.

وفى هذه الجلسة تقوم بأداء ما يلى :

- كتابة برنامج فى منقح بيسك السريع.
- تنفيذ البرنامج الذى كتبته.
- تتبع الاختيارات الأخرى للشاشة.
- حفظ البرنامج الذى كتبته.
- انتاج ملف تنفيذ EXE.
- الخروج من بيسك السريع.

إذا ما كان لهذا وقع بأنه عمل كبير فلا تهتم بذلك. فلن تستغرق الجلسة أكثر من 20 دقيقة فقط.

البـدء

يفترض أنك معتاد بدرجة كافية على تشغيل جهاز الكمبيوتر المتاح لك.

نظام قرصين مرنين : إذا كان لديك نظام قرصين فاستمر على النحو التالى :

١ - ادخل قرص المترجم رقم 1 فى المشغل A.

٢ - ادخل قرص العمل (قرص مشكل فارغ) فى المشغل B.

٣ - من المشغل B اكتب A:QB واضغط على مفتاح الإدخال.

نظام القرص الصلب : إذا كان لديك نظام قرص صلب (قرص ثابت) فاستمر على النحو التالى:

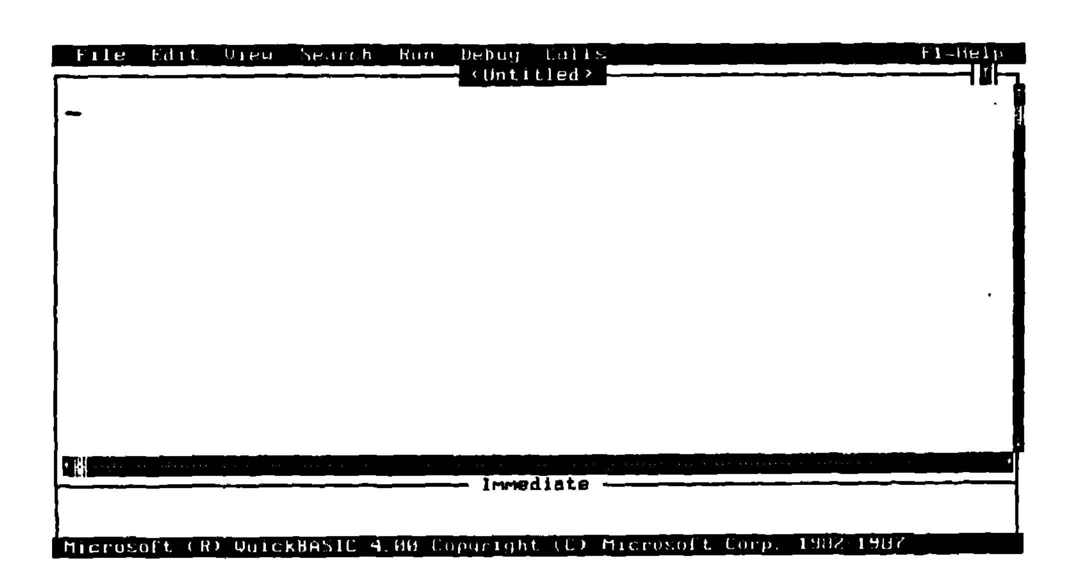
ملاحظة

إذا لم تكن قد انتجت دليل بيسك السريع بالفعل ونسخت كل ملفات البرامج فيه فأفعل ذلك الآن.

١ - غير الدليل المفتوح إلى دليل بيسك السريع وذلك بكتابة CD\QB والضغط على مفتاح الادخال.

٢ - اكتب QB واضغط على مفتاح الادخال.

لاحظ أن شاشة بيسك السريع تظهر فى عدة ثوان. ويكون منقح بيسك السريع متاحاً لك لإدخال البرنامج وتشغيله. لاحظ كذلك خيارات القائمة الموجودة فى قمة الشاشة والنافذة الفورية الموجودة فى قاعدة الشاشة ونقطة البداية الموجودة فى الركن العلوى الأيسر.



ادخال البرنامج : منقح بيسك السريع عبارة عن شاشة كاملة وحساس للمحتوى والتكوين ووسيلة تحكم ذاتية لكتابة البرامج. واستخدام المنقح هو أحد الأنشطة البسيطة. لإدخال البرنامج ابدأ فى كتابته ببساطة. واضغط على مفتاح الادخال عند الانتهاء من كل سطر. فإذا لم يسبق لك استخدام منقح شاشة كاملاً (مثل Wordstar) فقد تكون فى حاجة، على ذلك، للرجوع إلى الملحق B للتعرف على منقح بيسك السريع. اكتب الآن البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                                F1=Help
                                  BOX.BAS
  'Constants
  CONST ulx = 10, uly = 5, Lrx = 40, Lry = 10
  CONST horiz = 196, Vert = 179, trc = 191, tlc = 218, brc = 217, blc = 192

  'Variables
  Lin% = 1
  WStr$ = ""

  GOTO Start

DrawBox:
  LOCATE uly, ulx: PRINT CHR$(tlc)
  LOCATE Lry, ulx: PRINT CHR$(blc)
  LOCATE uly, Lrx: PRINT CHR$(trc)
  LOCATE Lry, Lrx: PRINT CHR$(brc)
  LOCATE uly, ulx + 1: PRINT STRING$(Lrx - ulx - 1, horiz);
  LOCATE Lry, ulx + 1: PRINT STRING$(Lrx - ulx - 1, horiz);
  FOR Cnt = uly + 1 TO Lry - 1
    LOCATE Cnt, ulx: PRINT CHR$(Vert)
    LOCATE Cnt, Lrx: PRINT CHR$(Vert)
  NEXT Cnt
  RETURN


WriteInBox:
  IF LEN(WStr$) > Lrx - ulx - 2 THEN GOTO WriteRet
  IF Lin% > Lry - uly - 2 THEN GOTO WriteRet

  Tb% = ((Lrx - ulx - 2) - LEN(WStr$)) / 2
  LOCATE uly + Lin%, ulx + Tb%: PRINT WStr$;

WriteRet:
  RETURN

Start:
  CLS
  GOSUB DrawBox
  WStr$ = "WELCOME TO"
  GOSUB WriteInBox
  Lin% = Lin% + 1
  WStr$ = "Illustrated QuickBASIC"
  GOSUB WriteInBox

END

                                 Immediate

Main: <Untitled>        Context: Program not running                  00054:004
```

تشغيل البرنامج : بعد انتهائك من كتابة البرنامج استمر على النحو التالى :

ملاحظة :

بالنسبة إلى تسلسل المفاتيح المتصل بشرطة (مثل Ctrl-C) اضغط على المفتاح الأول واستمر فى الضغط عليه أثناء الضغط على المفتاح الثانى.

١ – اضغط على Shift-F5 لتشغيل البرنامج. فإذا ما اكتشف بيسك السريع أى أخطاء فلن يعمل البرنامج. تحت هذه الظروف يزيد بيسك السريع من اضاءة سطر البرنامج الموجود به خطأ ويظهر نافذة على الشاشة تصف الخطأ. فإذا ما حدث ذلك لك فاستمر على النحو التالى :

أ – لاحظ الخطأ الموجود على الشاشة. اضغط على قضيب المسافات Spacebar للتخلص من نافذة رسالة الخطأ.

ب – افحص قائمة البرنامج الموجودة فى الدرس واكتب العبارة الصحيحة.

ج – اضغط على Shift-F5 لتشغيل البرنامج.

٢ – لاحظ مخرجات البرنامج على الشاشة.

```
        WELCOME TO
   Illustrated QuickBASIC




Press any key to continue
```

٣ – اضغط على قضيب المسافات للعودة إلى منقح بيسك السريع.

فحص القوائم : افحص بسرعة قوائم بيسك السريع الأخرى على النحو التالى :

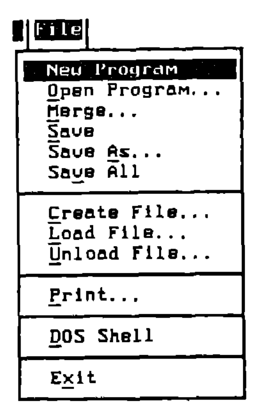
١ – اضغط على Alt-F لعرض قائمة الملفات.

٢ – اضغط على مفتاح السهم الأيمن لعرض القوائم الأخرى.

٣ – اضغط على Esc للعودة إلى المنقح.

حفظ البرنامج : احفظ برنامجك الآن بحيث يمكنك استخدامه مع سمات بيسك السريع الأخرى فيما بعد.

١ – اضغط على Alt-F للحصول على قائمة الملفات.



لاحظ الثلاث خيارات للحفظ. يحفظ خيار Save الملف بدون أى مزيد من اللغط. فإذا كان الملف جديداً فيسألك الخيار عن اسم الملف. خيار Save As يسمح لك بحفظ الملف تحت اسم جديد. سوف تستخدم هذا الخيار بكثرة فى دروس لاحقة. ويحفظ خيار Save All كل الملفات الموجودة حالياً فى الذاكرة.

٢ – اكتب A لاختيار Save As.

٣ – اكتب BOX.BAS كاسم للملف. لحفظ البرنامج على مشغل مختلف أضف مؤشر المشغل إلى اسم الملف. مثال ذلك B:BOX.BAS.

٤ – اضغط على Tab ومفتاح السهم السفلى لاختيار ملف نص ASCII.

٥ – اضغط على Tab مرة أخرى ثم على قضيب المسافات لحفظ الملف.

انتاج ملف تنفيذ EXE.: يمكن تنفيذ الملف الذى كتبته بدون استخدام بيسك السريع. لعمل ذلك يجب عليك أن تترجم هذا البرنامج وتحفظه على القرص. قائمة تنفيذ Run بيسك السريع يوجد بها اختيار يترجم البرنامج ويحفظه على القرص. والصيغة المترجمة يكون لها الاتساع EXE. ويمكن تنفيذ هذه الصيغة من DOS ببساطة عن طريق كتابة اسم الملف مثلما يلى :

```
A:>BOX
```

لتجربة ذلك استمر على النحو التالى :

١ – اضغط على Alt-R واختر الخيار Make EXE file.

٢ - لاحظ المستطيل الذى يظهر على الشاشة مع خيارات أكثر ووجود اسم الملف BOX.EXE الذى يكون هناك بالفعل. (اضغط على Tab ثلاث مرات ثم اضغط على قضيب المسافات. وينقلك هذا إلى الخيار Stand Alone EXE file.

٣ - اضغط على Tab مرة أخرى لاختيار الخيار Make EXE عند قاعدة المستطيل واضغط على قضيب المسافات.

يترجم بيسك السريع البرنامج ثم يصله ويكتبه على القرص. فإذا كانت لديك أى مشاكل أثناء هذه العملية فتأكد أن بيسك السريع قد سبق إعداده بطريقة صحيحة. ولمزيد من المعلومات عن إعداد بيسك السريع أرجع إلى الدرس الثانى.

لتشغيل ملف EXE من DOS استمر على النحو التالى :

١ - اضغط على Alt-F ثم اكتب D. يسمح لك هذا الخيار بتنفيذ أوامر DOS بدون وجود بيسك السريع.

٢ - اكتب BOX عند ملقن DOS واضغط على مفتاح الادخال. لاحظ أن البرنامج ينفذ.

٣ - اكتب Exit للعودة إلى بيسك السريع.

الخروج من بيسك السريع : عندما تنتهى من عملك كله وتريد الخروج من بيسك السريع لأداء أى عمل آخر استمر على النحو التالى :

١ - اضغط على Alt-F ثم اكتب X للخروج من منقح بيسك السريع.

٢ - انتقل إلى الدرس الرابع للاستمرار فى تسلسل التعلم.

الدرس الرابع

الترجمة من DOS

هناك طريقة أخرى لترجمة البرنامج الذى أعددته فى الدرس الثالث وتنفيذه وهى ترجمته من ملقن DOS باستخدام مترجم سطر الأمر BC.EXE. وتتكون هذه العملية من خطوتين :

١ - ترجمة

٢ - وتوصيل

أثناء خطوة الترجمة تترجم عبارات برنامج البيسك إلى لغة الآلة وتكتب فى ملف مع اسم البرنامج والتوسع OBJ. وفى خطوة التوصيل يضاف مقطع المكتبة المناسب إلى شفرة الآلة ويتم انتاج برنامج قائم بذاته وكتابته فى ملف له نفس اسم الملف ولكن بالتوسع EXE. ويمكن أن ينفذ هذا الملف بكتابة اسمه عند ملقن DOS والضغط على مفتاح الادخال.

للمزيد من المعلومات عن ترجمة سطر الأوامر وعملية الاتصال ارجع إلى الملحق D.

عندما تقوم بترجمة البرنامج BOX.BAS من بيسك السريع يجب أن تكون قد لاحظت أن بيسك السريع قد انتج العديد من أوامر DOS ونفذها. وفى هذا الدرس تقوم بكتابتها بنفسك وهى تنفذ.

وأسباب عمل ذلك بهذه الطريقة هى ما يلى :

- قد يكون البرنامج كبيراً جداً لترجمته من داخل بيسك السريع.
- قد تكون هناك حاجة إلى خيارات سطر الأوامر.
- يمكن أن يكتب البرنامج فى صورة مضغوطة بمصحح Code View.
- لانتاج برنامج ينفذ ومتوافق مع مصحح Code View.
- لانتاج قائمة ملف من الترجمة.
- لتوصيل NOCOM. OBJ الذى يقلل حجم ملفات EXE. التى لاتستخدم عبارة COM.

وقد تم تقديم ذلك هنا لإعطاء القارىء الشعور بالامكانيات المصاحبة لبيسك السريع. وسوف تقوم فى هذا الدرس بتنفيذ الأنشطة التالية بصورة مبسطة :

- ترجمة BOX.BAS من DOS باستخدام BC.EXE.
- توصيل BOX.OBJ وانتاج BOX.EXE باستخدام LINK.EXE.
- تنفيذ BOX.EXE.

الترجمة

تقوم فى هذه العملية بترجمة برنامج BOX.BAS مستخدماً مترجم بيسك السريع القائم بذاته. ابدأ عند ملقن DOS.

نظام قرص مرن واحد : إذا كنت تستخدم جهاز كمبيوتر به مشغل أقراص واحد فقط فاستمر على النحو التالى :

١ - ادخل قرص بيسك السريع واكتب BC ثم اضغط على مفتاح الادخال.
٢ - استبدل قرص البرنامج بالقرص الموجود عليه برامجك. اكتب BOX.BAS كاسم لملف المصدر واضغط على مفتاح الادخال. اضغط على مفتاح الادخال كإجابة على بقية الملقنات.

نظام قرصين مرنين : إذا كان لديك مشغلا أقراص فاستمر على النحو التالى :

١ - ادخل قرص بيسك السريع فى المشغل A وقرص برامجك فى المشغل B واجعل المشغل B هو المشغل المستخدم.
٢ - اكتب ;A:BC BOX.BAS/d/o واضغط على مفتاح الادخال.

نظام قرص صلب : إذا كان لديك قرص صلب فاستمر على النحو التالى :

١ - تأكد من وجودك فى دليل بيسك السريع بكتابتك CD\QB والضغط على مفتاح الادخال.
٢ - اكتب ;/BC BOX.BAS/d/o واضغط على مفتاح الادخال.

التوصيل

فى هذا القسم تقوم بتوصيل ملف OBJ، مع مكتبات لدعم وقت تنفيذ بيسك السريع. ويحدث ذلك طبقاً للطريقة التالية :

نظام قرص مرن واحد : يحدث التوصيل مع قرص مرن واحد فقط على النحو التالى :

١ - انسخ الملفات التالية على قرص مشكل وفارغ :

- BCOM40.LIB
- BOX.OBJ

٢ - ادخل قرص بيسك السريع فى المشغل واكتب LINK واضغط على مفتاح الادخال.

٣ - استبدل قرص بيسك السريع بالقرص الذى انتجته فى الخطوة الأولى. اكتب BOX واضغط على مفتاح الادخال ثم اضغط على مفتاح الادخال مرة أخرى كرد على بقية الملقنات.

نظام بمشغلى أقراص مرنة : يتم التوصيل بمشغلين على النحو التالى :

١ - إعداد قرص كما هو مذكور فى الخطوة الأولى من قسم المشغل الواحد.

٢ - ادخل قرص بيسك السريع فى المشغل A والقرص الناتج فى الخطوة الأولى فى المشغل B.

اجعل المشغل B هو المشغل المستخدم وذلك بكتابة : B والضغط على مفتاح الادخال.

٣ - اكتب A: LINK BOX.OBJ, BOX.EXE ثم اضغط على مفتاح الادخال.

تنفيذ البرنامج

لتنفيذ البرنامج الذى سبق لك ترجمته اتبع ما يلى :

نظام قرصين مرنين:

١ - ادخل القرص الموجود فيه "BOX. EXE" فى المشغل A.

٢ - غير المشغلات إلى A واكتب BOX واضغط على مفتاح الادخال.

نظام قرص صلب:

١ - إذا لم تكن فى الدليل "C: \QB" فانتقل إليه.

٢ - اكتب BOX واضغط على مفتاح الإدخال.

انتقل إلى الدرس الثالث والعشرين للاستمرار فى تسلسل التعلم.

الدرس الخامس

دالة القيمة المطلقة ABS

الوصف

تعيد دالة القيمة المطلقة ABS قيمة التعبير العددى المطلقة وتكوينها هو كما يلى :

ABS (تعبير عددى)

وتعود قيمة التعبير العددى كقيمة صحيحة بدون اشارة.

التطبيقات

تستخدم دالة ABS فى الحصول على الجزء الصحيح لتعبير عددى بدون اشارة. فى بعض الأحيان لا تكون اشارة العدد مهمة فى اتخاذ القرارات وتصبح دالة ABS مفيدة فى هذه الأحوال. وفيما يلى بعض الأمثلة.

مثال ١

```
X = 38.3: Y = -99
Z = ABS(X-Y)
```

مثال ٢

```
QuadRoot# = 1.316074          ' square root of square root of 3
PRINT ABS(QuadRoot#)
```

عملية تقليدية

توضح دالة ABS فى العملية التقليدية التالية. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                                F1=Help
                              <Untitled>
'Program to iteratively take the square root until the result approaches zero
PRINT "Enter a number to get the square root."
PRINT "A negative number ends the program."

INPUT "Enter number "; x
IF x <= 0 THEN END
Iter% = 1: PSqr = SQR(x)
DO WHILE ABS(1 - PSqr) >= .00001
  PSqr = SQR(PSqr)
  Iter% = Iter% + 1
LOOP
PRINT "The square root of the number was taken until it approached zero."
PRINT "Number of iterations: "; Iter%

──────────────────────────── Immediate ────────────────────────────

Main: <Untitled>        Context: Program not running                 00013:008
```

٢ – نفذ البرنامج ولاحظ استخدام دالة ABS فى البرنامج.

```
Enter a number to get the square root.
A negative number ends the program.
Enter number ? 233
The square root of the number was taken until it approached zero.
Number of iterations:  20









Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر Save من قائمة الملفات واكتب ABS.BAS كاسم للملف وحدد أن صيغة الملف نصية واحفظ البرنامج.

٤ – من قائمة الملفات اختر New مع اخلاء الشاشة.

٥ – انتقل إلى الدرس المائة والأربعين للاستمرار فى تسلسل التعلم.

الدرس السادس

دالة ASC

الوصف

تعيد دالة ASC قيمة ASCII لأول رمز من تعبير السلسلة، وتكوينها هو ما يلي :

ASC (تعبير سلسلة)

إذا كان تعبير السلسلة فارغاً فينتج المترجم رسالة خطأ وقت التشغيل.

التطبيقات

دالة ASC مفيدة جداً في الحصول على قيمة ترتيبية من جدول ASCII لرمز معين. ويمكن أن تستخدم ضمن أشياء أخرى في تحويل تعبيرات السلاسل للأعداد إلى قيم عددية ثم تحويل السلاسل بعد ذلك إلى الحروف العلوية (الكبيرة) أو السفلية (الصغيرة).

وتعيد الدالة : `ASC("A")`

قيمة ASCII للحرف "A"

كما تعيد الدالة : `ASC("Hello")`

قيمة ASCII للحرف "H"

والبرنامج :

```
Exp$ = "Prep"
FOR Cnt = 1 TO LEN(Exp$)
     PRINT ASC(MID$(Exp$,Cnt,1)) " ";
NEXT Cnt
```

يطبع قيمة ASCII لكل رمز موجود في السلسلة Exp$.

عملية تقليدية

يحول المثال التالي أحد تعبيرات السلسلة إلى قيمة عددية. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالي :

```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                 <Untitled>
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
  Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%

──────────────────────────── Immediate ────────────────────────────

Main: <Untitled>      Context: Program not running                 00005:017
```

٢ – نفذ البرنامج ولاحظ تحويل السلسلة Num$ إلى مكافئها العددى. لاحظ كذلك استخدام قيمة ASCII الناتجة من دالة ASC

```
1234          1234





Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج.

٤ - اختر Save من قائمة الملفات واكتب ASC.BAS كاسم للملف. حدد أن تشكيل الملف نصى واحفظ الملف.

٥ - انتقل إلى الدرس الثانى والستين للاستمرار فى تسلسل التعلم.

الدرس السابع

دالـة ATN

الوصف

تعيد دالة ATN قوس الظل لتعبير عددى. وقوس الظل هو الزاوية التى يكون ظلها مساوياً للتعبير العددى، وتكوينها هو كما يلى :

ATN (تعبير عددى)

تكون القيمة الناتجة بالتقدير الدائرى وتقع فى العدى من $\pi/2$ - إلـى $\pi/2$ + (حـيث $\pi = 3.141593$، وتقوم هذه القيمة كعدد له دقة فردية كاصطلاح تقليدى أما إذا كان التعبير العددى له دقة مزدوجة فتعود قيمة قوس الظل بدقة مزدوجة. ويكون التعبير العددى من أى نوع من الأنواع العددية.

التطبيقات

تستخدم دالة ATN عندما تكون هناك حاجة إلى قوس الظل لأحد الأعداد كما فى حالة تشغيل الاعداد أو تطبيقات الألعاب. وفيما يلى بعض الأمثلة :

```
Pi = 3.141592653
At = ATN(Pi/1.5)
PRINT ATN(Pi/3)
```

عملية تقليدية

توضح العملية التالية دالة ATN. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the ATN function.
CLS
PRINT "The program computes polar coordinates from rectangular"
PRINT "coordinates."
PRINT
INPUT "Enter x and y separated by commas "; x, y
IF x + y = 0 THEN GOTO 111
IF x = 0 THEN GOTO 222
IF y = 0 THEN GOTO 333

AngleDeg = ATN(y / x)
AngleDeg = (AngleDeg * 180) / 3.14159
Mag = SQR(x ^ 2 + y ^ 2)
GOTO PrintLine

111 :
  AngleDeg = 0: Mag = 0
  GOTO PrintLine

222 :
  IF y > 0 THEN
    AngleDeg = ABS(y)
  ELSE
    AngleDeg = -90
    Mag = ABS(y)
  END IF
  GOTO PrintLine

333 :
  IF x > 0 THEN
     AngleDeg = 0: Mag = x
  ELSE
     AngleDeg = 180
     Mag = ABS(x)
  END IF

PrintLine:
  PRINT "Polar coordinates: Angle in degrees = "; AngleDeg
  PRINT "Magnitude = "; Mag

------------------------- Immediate -------------------------

Main: <Untitled>         Context: Program not running              00017:023
```

٢ – نفذ البرنامج. لاحظ النتائج واستخدام دالة ATN في البرنامج.

```
The program computes polar coordinates from rectangular
coordinates.

Enter x and y separated by commas ? 10,10
Polar coordinates: Angle in degrees =  45.00004
Magnitude =  14.14214







Press any key to continue
```

٣ – اضغط على أى مفتاح وعد إلى البرنامج. اختر Save من قائمة الملفات واكتب ATN.BAS كاسم للملف وحدد أن صيغة الملف نصية واحفظ البرنامج.

٤ – انتقل إلى الدرس الرابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس الثامن

عبارة BEEP

الوصف

تنتج عبارة BEEP صوتاً من مكبر الصوت الموجود في الكمبيوتر. وتكوينها هو كما يلي :

BEEP

التطبيقات

عبارة BEEP مفيدة في شد انتباه المستفيد إلى النشاط الحالي في البرنامج. وفيما يلي بعض الأمثلة :

```
IF (INKEY$ = "") THEN BEEP
IF (Cnt% > 3) THEN BEEP
IF ErrSignal THEN BEEP
BEEP:BEEP:BEEP: PRINT "No more records !"
```

عملية تقليدية

توضح هذه العملية استخدام عبارة BEEP. ابدأ بتحميل بيسك السريع.

١ – أكتب البرنامج التالي :

```
File  Edit  View  Search  Run  Debug  Calls                F1=Help
                         <Untitled>
'This program demonstrates the BEEP statement
CLS
Sel1$ = "Selection 1": Sel2$ = "Selection 2": Sel3$ = "Selection 3"
LOCATE 2, 30: PRINT "MENU SELECTION"
LOCATE 4, 30: PRINT Sel1$
LOCATE 5, 30: PRINT Sel2$
LOCATE 6, 30: PRINT Sel3$
LOCATE 11, 30: PRINT "Choose .."

LOCATE 11, 45: INPUT C$
IF C$ < "1" OR C$ > "3" THEN BEEP

                         Immediate

Main: <Untitled>    Context: Program not running      00011:019
```

٢ - نفذ البرنامج. اكتب 4 واضغط على مفتاح الادخال. لاحظ استخدام عبارة BEEP فى البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

```
MENU SELECTION

Selection 1
Selection 2
Selection 3



Choose ..       ? 4









Press any key to continue
```

٣ - اختر New من قائمة الملفات واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والثانى والثلاثين للاستمرار فى تسلسل التعلم.

الدرس التاسع

عبارتا BLOAD و BSAVE

الوصف

عبارة BLOAD : تقوم العبارة بتحميل صورة من الذاكرة للملف أو لوحدة من فرع محدد من الذاكرة. وتكوينها هو كما يلي :

```
BLOAD file spec, offset
```

جزء file spec هو نفسه مثل مواصفات الملف المقدمة للأوامر الأخرى لتشغيل الملفات مثل FILES و KILL و NAME AS. وتوضع أسماء المسارات وأسماء الملفات بين علامات تنصيص وتتبع في تكوينها نفس التكوين الذي تتبعه أسماء المسارات وأسماء الملفات في نظام تشغيل DOS. وجزء offset هو فرع القطاع الذي سوف تحمل البيانات فيه. والقطاع هو آخر DEF SEG ينفذ أو قطاع بيسك السريع كحالة تقليدية.

عبارة BSAVE : تنسخ العبارة البيانات من الذاكرة كما هي تماماً في ملف محدد أو مشغل معين. وتكوينها هو كما يلي :

```
BSAVE file spec, offset, length
```

جزءا file spec و offset هما مثل نظيريهما في العبارة السابقة ويحدد جزء length عدد البايت الذي ينسخ من الذاكرة، وهو رقم يقع بين 0 و 65,535. وعلى هذا فالملف الذي ينتج هو نسخة من الذاكرة بايت بعد بايت مع معلومات تحكم بيسك السريع.

كل وحدات المدخلات باستثناء لوحة المفاتيح تدعم بواسطة هاتين العبارتين. ولاتدعم العبارتان من وحدة تشغيل الشرائط.

التطبيقات

هاتان العبارتان هما من العبارات المطورة المستخدمة في التعامل مع ذاكرة وقت تنفيذ بيسك السريع. ويجب استخدامهما بكل حذر. ولا تختبر عبارة BLOAD الذاكرة التي تنقل البيانات إليها وبالتالي فمن الممكن أن تكتب على البيانات الموجودة قبل النقل. وإذا ما حدث ذلك فتكون النتائج غير متوقعة. عند تحميل شاشات من ملف إلى داخل الذاكرة فمن الضروري أن تكون

حالة الشاشة هى نفسها مثلما كانت عليه عندما حدث حفظ للشاشة. أى أنه إذا كانت الشاشة قد انتجت وحفظت وهى فى حالة الرسومات فيجب أن تكون فى نفس حالة الرسومات كذلك عندما يتم تحميلها مرة أخرى. ويجب ألا تحمل ملفات سبق انتاجها باستخدام بيسك المطور BASICA وذلك بعبارة BLOAD لأن بيسك السريع يحمل البرامج ويخزن البيانات فى مواقع مختلفة عما يفعله بيسك المطور. وفيما يلى أمثلة للعبارتين :

مثال١

```
DIM W(10,10)
..
DEF SEG = VARSEG(W(1))
BSAVE "W.Dat", VARPTR(W(1)), 404
```

مثال٢

```
DEF SEG = &hb000
..
BLOAD "Screen.001"
```

المثال السابق يحمل صورة للشاشة داخل ذاكرة الشاشة. ويمكن أن تستخدم هذه الطريقة فى تحميل شاشات سبق انتاجها باستخدام برامج أخرى، ويقلل هذا من الشفرة اللازمة لانتاج الشاشة أثناء وقت التنفيذ.

عملية تقليدية

تستخدم هذه العملية العبارتين السابقتين فى حفظ وتحميل منظومة عددية. يتم انتاج المنظومة ووضع البيانات فيها كقيم ابتدائية ثم تخزن المنظومة فى ملف ثم تمحى ويعاد عمل الأبعاد لها ثم تحمل من الملف. ويطبع عدد محدود من عناصر المنظومة لتوضيح أنها تعمل.

ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                                   F1=Help
                                    BLSAVE.BAS
'This is a demonstration of the BLOAD and BSAVE statements.

'$DYNAMIC
DIM w(10, 10)
DEF SEG = VARSEG(w(1, 1))
FOR i = 1 TO 10
  FOR iZ = 1 TO 10
    w(i, iZ) = i + 1Z
  NEXT iZ, i

'Note the length provided is the number of elements in array w plus 1
'multiplied by 4.

BSAVE "W.Dat", VARPTR(w(1, 1)), 404
FOR i = 1 TO 10
  FOR iZ = 1 TO 10
    w(i, iZ) = 0
NEXT iZ, i
ERASE w
REDIM w(10, 10)

BLOAD "W.Dat", VARPTR(w(1, 1))
CLS

FOR i = 1 TO Z
  FOR iZ = 1 TO 10
    PRINT (w(i, iZ))
NEXT iZ, i

END

                                    Immediate

Main: BLSAVE.BAS        Context: Program not running                  00001:001
```

٢ – نفذ البرنامج. لاحظ استخدام عبارتى هذا الدرس. وتكون المخرجات على النحو التالى :

```
 13
 13
 13
 13
 13
 13
 13
 13
 13
 13
 14
 14
 14
 14
 14
 14
 14
 14
 14
 14

Press any key to continue
```

٣ - ارجع إلى البرنامج. احفظ البرنامج كبرنامج نصى تحت اسم BLSAVE.BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس الخمسين للاستمرار فى تسلسل التعلم.

الدرس العاشـر

عبارتا CALL و CALLS

الوصف

عبارة CALL : تستدعى العبارة برنامجاً فرعياً مكتوباً بالبيسك السريع أو بأى لغة برمجة أخرى. وتكوينها هو كما يلى :

(قائمة البرنامج الفرعى) اسم البرنامج الفرعى CALL

اسم البرنامج الفرعى هو اسم البرنا ج الفرعى المراد استدعاؤه. ويمكن أن يكتب الاسم بحد أقصى 40 خانة. وقائمة البرنامج الفرعى هى البيانات التى تمرر إلى مؤشرات البرنامج الفرعى. ويجب أن توضع أقواس حول هذا الجزء إلا إذا ما حذفت كلمة CALL وفى هذه الحالة يصبح تكوين العبارة على النحو التالى :

(قائمة البرنامج الفرعى) اسم البرنامج الفرعى

ويجب أن تكون القائمة من نفس النوع ولها نفس عدد مؤشرات البرنامج الفرعى.

عندما تستخدم عبارة CALL فى استدعاء برامج فرعية مكتوبة بلغات أخرى فتكون للقائمة تكوين مختلف كما يلى :

الكلمات الرئيسية BYVAL و SEG تعرف كيفية تمرير القائمة إلى البرنامج الفرعى، فتحدد BYVAL أن القائمة تمرر بالقيمة. وتحدد SEG أن القائمة تمرر كعنوان. ويجب أن تستخدم SEG بحذر عند تمرير المنظومات وذلك لأن بيسك السريع قد ينقل متغيرات فى الذاكرة قبل نقل التحكم إلى البرنامج الفرعى.

عندما تستخدم عبارة CALL بدون كلمة CALL فيجب أن توجد عبارة DECLARE لتمكن من التأكد من نوع القائمة.

عبارة CALLS : استخدام عبارة CALLS هو نفسه مثل استخدام عبارة CALL مع تمرير قائمة من نوع SEG إلى البرنامج المنادى. وتستخدم عبارة CALLS فى استدعاء برامج فرعية مكتوبة بلغات أخرى غير لغة بيسك السريع. ويتم تمرير القوائم كعناوين فقط. وتكوينها هو كما يلى :

(قائمة البرنامج الفرعى) اسم برنامج فرعى CALLS

واسم البرنامج الفرعى هو اسم البرنامج الفرعى المراد استدعاؤه وتقع عليه نفس القيود التى تقع على اسم البرنامج الفرعى المستخدم فى عبارة CALL التى سبق ذكرها .

التطبيقات

تستخدم عبارات CALL و CALLS فى استدعاء برامج فرعية. وعبارات الاستدعاء CALL و CALLS تستخدم أساساً فى البرمجة بلغات مختلفة. وعند استدعاء برامج فرعية مكتوبة ببيسك السريع توفر عبارة CALL آلية مرنة لضمان المعلومات التى تمرر إلى الاجراء المنادى عليه تكون من النوع الصحيح. وعند غياب كلمة CALL من عبارة CALL فتستخدم عبارة DECLARE لاجراء التأكد من نوع القائمة. وفيما يلى بعض الأمثلة :

مثال١

```
CALL ShowMenu
..
SUB ShowMenu;
      ..
      ..
END SUB
```

مثال٢

```
DECLARE SetUp (Q1 AS INTEGER)
..
SetUp Q1
..
SUB SetUp(Q1 AS INTEGER)
      ..
      ..
END SUB
```

عملية تقليدية

يتعامل البرنامج المستخدم فى هذه العملية مع عبارة CALL التى تستدعى برامج فرعية مكتوبة ببيسك السريع. وقد أعد البرنامج فى الدرس الواحد والسبعين. ابدأ بتحميل بيسك السريع.

١ - حمل البرنامج الموجود فى الدرس الواحد والسبعين والمسمى LUBOUND.BAS.

٢ - عدل عبارات CALL إلى SUB كما هو موضح فيما يلى (كلمة CALL نفسها لم تستخدم):

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                               LUBOUND.BAS
'The following program demonstrates the use of the UBOUND and LBOUND
'statements. The program loads two sets of array values and
'finds the minimum and maximum values in those arrays.

DECLARE SUB FindMinMax (A%(), MinVal!, MaxVal!)

Max = 15
DIM A%(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A%(Cnt)
  NEXT
  RETURN


Start:
  CLS
  GOSUB LoadArray
  PRINT "First pass"
  FindMinMax A%(), MinVal, MaxVal
  READ Max
  REDIM A%(Max)
  GOSUB LoadArray
  PRINT "Second pass"
  FindMinMax A%(), MinVal, MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,88,67,54,18
DATA 10
DATA 8,89,76,54,23,32,12,4,33,54
```

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                          LUBOUND.BAS:FindMinMax
SUB FindMinMax (A%(), MinVal, MaxVal)
 MinVal = A%(1): MaxVal = A%(1)
 FOR Cnt = LBOUND(A%) + 1 TO UBOUND(A%)
   IF MinVal > A%(Cnt) THEN
      MinVal = A%(Cnt)
   END IF
   IF MaxVal < A%(Cnt) THEN
      MaxVal = A%(Cnt)
   END IF
 NEXT
 PRINT "Minimum value in array: "; MinVal, "Maximum value in array: "; MaxVal
END SUB



──────────────────────────── Immediate ────────────────────────────

 Main: LUBOUND.BAS     Context: Program not running              00001:001
```

٣ - نفذ البرنامج ولاحظ استخدام عبارة CALL فى البرنامج، لاحظ غياب كلمة CALL من عبارة CALL وغياب الأقواس من حول القائمة.

٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٥ - انتقل إلى الدرس الحادى عشر للاستمرار فى تسلسل التعلم.

الدرس الحادى عشر

عبارة CALL ABSOLUTE

الوصف

تنقل هذه العبارة تنفيذ البرنامج إلى برنامج فرعى مكتوب بلغة الآلة. وتكوينها هو كما يلى :

CALL ABSOLUTE (متغير صحيح، قائمة)

القائمة هى القائمة الاختيارية التى تمرر إلى البرنامج الفرعى المكتوب بلغة الآلة. والمتغير الصحيح هو فرع الجزء المكتوب بلغة الآلة والذى ينتقل إليه التحكم. ويحدد القطاع باستخدام عبارة DEF SEG. والقوائم التى تمرر إلى البرنامج الفرعى عبارة عن مشيرات قريبة أو فروع من داخل القطاع الحالى.

ملاحظة

يجب أن يحمل بيسك السريع مع المكتبة السريعة "QB.QLB" لكى يمكن لهذه العبارة أن تعمل. (اكتب QB\LQB عند ملقن DOS واضغط على مفتاح الادخال).

التطبيقات

هذه العبارة توجد فى بيسك السريع للحفاظ على بعض التوافقية مع صيغ بيسك القديمة. فعبارات CALL و CALLS والتى تستدعى برامج فرعية مكتوبة بلغات أخرى تمثل طريقة سهلة لاستدعاء برامج فرعية مكتوبة بلغة المجمع. والبرامج الفرعية المكتوبة بلغة المجمع التى سبق اعدادها باستخدام بيسك المطور يجب أن تعدل إذا ما كانت تقبل قوائم سلاسل وذلك لأن واصفات السلاسل يكون طولها 4 بايت فى بيسك السريع. ويوجد مثال لهذه العبارة فى جزء البرنامج التالى :

```
DIM FastPrint(7)
'Get the offset of the array where the assembly routine will be placed.
Ptr = VARPTR(FastPrint(1))
'Set the segment to the segment of the array.
DEF SEG = VARSEG(FastPrint(1))
'Poke the instructions in memory beginning with the offset of the array.
FOR i = 1 to 40
      READ Byte
      POKE (Ptr + 1), Byte
NEXT
'Execute the call with a parameter T.
CALL ABSOLUTE(T,VARPTR(FastPrint(1)))
```

تظهر تعليمات المقطع المكتوب بلغة الآلة فى هذا البرنامج فى عبارة DATA كقيمة ممثلة بالنظام السادس عشرى.

عملية تقليدية

حيث أنه لا يمكن افتراض أن القارىء ملم بالبرمجة بلغة المجمع لايكون للعملية التقليدية أى معنى عملى فى هذا الدرس.

انتقل إلى الدرس الثانى عشر للاستمرار فى تسلسل التعلم.

الدرس الثانى عشر

عبارتا CALL86OLD و CALL INTERRUPT

الوصف

تنفذ العبارتان وظيفة استدعاء نظام التشغيل DOS من داخل بيسك السريع. وتكوين كل منهما هو كما يلى :

```
CALL  INT86OLD(intr, in_array, out_array)
CALL  INT86XOLD(intr, in_array, out_array)

CALL  INTERRUPT(intr, inregs, outregs)
CALL  INTERRUPTX(intr, inregs, outregs)
```

لاحظ أن مؤشرات كل من الأربعة تكوينات هى نفس المؤشرات. يحدد جزء intr رقم ازعاج DOS. وجزئى in-array و out-array فى تكوينات INT86OLD و INT86XOLD هما منظومتان عدديتان صحيحتان فى نوعهما مع ثمانية عناصر لعبارة INT86OLD وعشرة عناصر لعبارة INT86XOLD. وعناصر المنظومة هى كما يلى :

```
in_array(1)          Register AX
in_array(2)              "    BX
in_array(3)              "    CX
in_array(4)              "    DX
in_array(5)              "    BP
in_array(6)              "    SI
in_array(7)              "    DI
in_array(8)              "    Flags
```

كما أن المنظومة تشمل كذلك العنصرين التاليين فى INT86XOLD :

```
in_array(9)              "    DI
in_array(10)             "    Fl
```

وتتبع المنظومة out-array نفس التكوين. وكل المسجلات باستثناء BP و DS تعدل بواسطة الاستدعاءات.

وفى تكوين CALL INTERRUPT تكون الاعداد الصحيحة والمنظومة على الصورة التالية :

```
TYPE   RegType
     AX AS INTEGER
     BX AS INTEGER
     CX AS INTEGER
     DX AS INTEGER
     BP AS INTEGER
     SI AS INTEGER
     DI AS INTEGER
     FLAGS AS INTEGER
     DS AS INTEGER
     ES AS INTEGER
END TYPE
```

ويوجد توضيح النوع فى ملف الشمول "Qb. Bi" وهو ضرورى لاستخدام وظائف الاستدعاء هذه.

يحتوى جزء in-array فى كل عبارات CALL السابقة على قيم مسجلات تستخدم فى تنفيذ ازعاج DOS ويحتوى جزء out-array على قيم المسجلات بعد حدوث الازعاج.

وتوزع دوال INT86OLD و INT86XOLD و INTERRUPT و INTERRUPTX كمكتبة سريعة مع المترجم عندما يتم تنفيذ بيسك السريع مع خيار سطر الأمر 1qb/. ووظائف الازعاج هى وسائل برمجة مطورة تستخدم فى الحصول على معلومات من DOS وأداء ازعاجات DOS مباشرة. والإزعاجات interrupts هى أنشطة تزعج أى شئ آخر يجب أن يحدث لأداء وظيفة محددة من وظائف DOS. ويأخذ العديد من المقاطع مثل هذه الكتابة إلى الشاشة، مع اختيار حالة الرؤية video، وانتاج ملف ينفذ على أنه ازعاجات. إلا أن هذه تنتقل إلى المستفيد وعلى هذا فلايحتاج المستفيد أن يهتم بها.

ملاحظة

يجب أن يحمل بيسك السريع مع المكتبة السريعة "QB. QLB" لكى يمكن لهذه العبارة أن تعمل. (اكتب QB\LQB عند ملقن DOS واضغط على مفتاح الادخال).

التطبيقات

تصبح استدعاءات وظيفة DOS ضرورية عندما لا يمكن تنفيذ عملية معينة بوسيلة التطوير المستخدمة. احدى هذه المواقف يمكن أن يكون ايجاد معلومات عن النظام فى بداية البرنامج مثل

ما إذا كان هناك مطبع adaptor للرسومات الملونة أو لا أو ما هو حجم الذاكرة المتاحة. ويجب أن تستخدم ازعاجات DOS إذا لم يكن هناك أى طريقة أخرى لأداء هذه الأنشطة فقط. وفيما يلى أمثلة لعبارات الازعاج CALL :

مثال١

```
'Include the header file from QuickBASIC.
'$INCLUDE:'Qb.Bi'

DIM in_array(7), out_array(7)
in_array(1) = ..
in_array(3) = ..
in_array(4) = ..
intr        = ..

CALL INT86OLD(intr,in_array(),out_array())
```

مثال٢

```
DIM inregs AS RegType, outregs AS RegType

inregs.AX = ..: inregs.CX = ..: inregs.DX = ..
intr = ..

CALL INTERRUPT(intr,inregs,outregs)
```

وتقدم وظائف INT86OLD و INT86XOLD للحفاظ على التوافقية مع صيغ البيسك الأخرى.

عملية تقليدية

هذه العملية توضح عبارة CALL INTERRUPT. ابدأ عند ملقن DOS.

١ - اكتب QB\LQB واضغط على مفتاح الادخال. (ارجع إلى الملحق B للمزيد من المعلومات عن تحميل المكتبات السريعة).

٢ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                 CALLINT.BAS
'test the DOS interrupt procedures
'$INCLUDE: '\qb\QB.BI'
CLS

DIM InRegs AS RegType, OutRegs AS RegType
FOR c = 1 TO 2000
  PRINT "x";
NEXT
DO
LOOP UNTIL INKEY$ <> ""

'set AX for scroll down function
InRegs.ax = &H700
'set start row,col
InRegs.cx = &H0
'set stop row,col
InRegs.dx = &HF40
'call int 10
CALL INTERRUPT(&H10, InRegs, OutRegs)

                                 Immediate

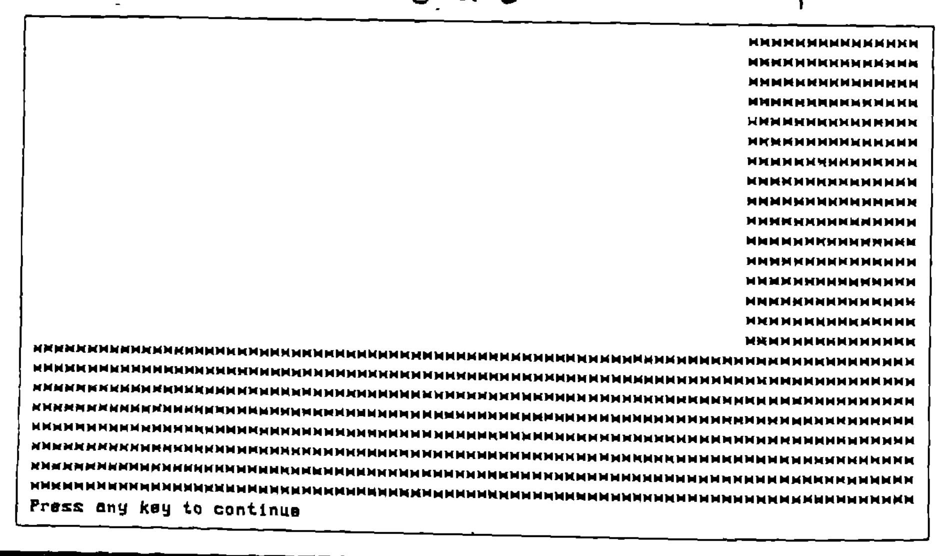
Main: CALLINT.BAS    Context: Program not running                 00036:001
```

ملاحظـة

فى عـبارة $INCLUDE يكون المسـار المقـدم مـحـدداً لتكوين دليل الكمـبيـوتر. اسـتبـدل "\qb\QB.B1" بالمسار المناسب لجهاز الكمبيوتر المتاح لك.

٣ – نفذ البرنامج وأضغط على مفتاح الإدخال. المخرجات تشبه ما يلى :

لاحظ استخدام CALL INTERRUPT فى البرنامج.



٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٥ - انتقل إلى الدرس الرابع عشر للاستمرار في تسلسل التعلم.

الدرس الثالث عشر

دوال CDBL و CINT و CLNG و CSNG

الوصف

دالـة CDBL : هذه الدالـة تحول التعبير العددى إلى قيمة مزدوجة الدقـة. وتكوينها هـو كمايلـى :

(تعبير عددى) CDBL

التعبير العددى هـو القيمة التى تتحول إلى عدد له دقة مزدوجة. ولا يعنى ذلك أن النتيجة تكون أكثر دقـة من العـدد الأصلى بالرغم من وجود أرقـام دقة أكثر. وللدالة نفس التـأثير مـثل تحديد تعبير عددى بأنه متغير مزدوج الدقة.

دالة CINT : هذه الدالة تحـول التـعبير العددى إلى قيمة صحيحة عن طريق التقريب. وتكوينها هو كما يلى :

(تعبير عددى) CINT

التعبير العددى هو القيمة التى يحدث لها تقريب وتعود كقيمة عددية صحيحة. ويجب أن تقع القيمة العددية بين 32,768- و 32,767. وعندما لا يتحقق ذلك تظهر رسالة السريان الزائد. وعلى عكس دالتى FIX و INT اللتين تحذفان الكسر كاملاً فإن دالة CINT تقرب القيمة.

دالة CLNG : هذه الدالة تحـول التعبير العددى إلى قيمة صحيحة طويلة. وتكوينها هو كمايلى :

(تعبير عددى) CLNG

تتحول قيمة التعبير العددى إلى عدد صحيح طويل (4 بايت). وعندما لاتقع قيمة التعبير العددى فى المدى من 2,147,483,648- إلى 2,147,483,647 فتظهر رسالة السريان الزائد.

دالة CSNG : هذه الدالة تحول التعبير العددى إلى قيمة فردية الدقة. وتكوينها هو كمايلى:

(تعبير عددى) CSNG

تحول القيمة العددية إلى عدد له دقة فردية. وهذا يشبه تحديد تعبير عددى لمتغير له دقة فردية. ويتم تقريب هذه القيمة إذا ما كانت هناك حاجة إلى ذلك.

التطبيقات

دوال CDBL و CINT و CLNG و CSNG هى جزء من ترسانة اجراءات تحويل النوع. وتستخدم هذه الدوال فى تحويل قيم عددية من نوع إلى آخر عندما تكون هناك حاجة إلى ذلك. وفيما يلى بعض الأمثلة :

مثال١

```
DEFSNG q-t: DEFDBL l-p
Qrt = 32.01/.021
Press = CDBL(Qrt)
```

مثال٢

```
DEFINT m-p: DEFLNG r-t
Mts = 32 * 1000
Rts = CLNG(Mts)
```

توضح الأمثلة استخدام عملية تحويل الأنواع العددية.

عملية تقليدية

توضح هذه العملية استخدام دوال CDBL و CINT و CLNG و CSNG. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the use of the CSNG, CINT, CDBL and CLNG
'functions. The program assigns values to variables defined to
'be of other types.

DEFINT I-L: DEFSNG M-P: DEFDBL Q-T
DEFLNG A-D

I = 128: L = 256
M = 22 / 7
Q = 22 / 7
A = 256000

CLS
PRINT "DEFINT vars. "; I, L
PRINT "DEFSNG vars. "; M
PRINT "DEFDBL vars. "; Q
PRINT "DEFLNG vars. "; A

I = CINT(A / 10)
A = CLNG(10! * 10)
M = CSNG(Q)
Q = CDBL(13 / 2)

PRINT "After CINT "; I
PRINT "After CLNG "; A
PRINT "After CSNG "; M
PRINT "After CDBL "; Q

                              Immediate

Main: <Untitled>      Context: Program not running          C  00036:023
```

٢ – نفذ البرنامج. لاحظ استخدام دوال CDBL و CINT و CLNG و CSNG فى البرنامج.

```
DEFINT vars.  128          256
DEFSNG vars.  3.142857
DEFDBL vars.  3.142857074737549
DEFLNG vars.  256000
After CINT  25600
After CLNG  100
After CSNG  3.142857
After CDBL  6.5

Press any key to continue
```

٣ – ارجع إلى البرنامج مع إخلاء الشاشة دون أن تحفظ البرنامج.

٤ – انتقل إلى الدرس الثلاثين للاستمرار فى تسلسل التعلم.

الدرس الرابع عشر

عبارة CHAIN

الوصف

تتسبب عبارة CHAIN فى تنفيذ برنامج بيسك سريع من برنامج بيسك سريع آخر. وتكوينها هو كما يلى :

```
CHAIN file spec
```

جزء file spec هو اسم ملف البرنامج المراد تنفيذه. ولايعود التحكم إلى البرنامج المنادى بعد تنفيذ البرنامج المطلوب. والبرامج التى تنفذ فى بيئة تطوير بيسك السريع لايمكنها أن تستدعى برامج بيسك قائمة بذاتها (EXE.). والبرامج التى تنفذ على أنها برامج قائمة بذاتها لاتستطيع أن تستدعى ملفات مصدر بيسك السريع. ويمكن أن تستخدم المتغيرات فى البرنامجين وتمرر إلى البرنامج المنادى عليه من خلال مجموعة مشاركة COMMON فارغة.

التطبيقات

تستخدم عبارة CHAIN فى استدعاء برامج بيسك السريع من داخل البرنامج. والبرنامج المنادى عليه يمكن توصيله بالبرنامج الأصلى وحيث إن عبارة CHAIN لا تقبل أرقاماً للأسطر كجزء من مواصفة الملف فيجب أن تكتب عملية التسلسل بعناية فائقة لتجنب عمل الدورات اللانهائية. وفيما يلى بعض الأمثلة لعبارة CHAIN :

مثال١

```
IF Proc > 2 THEN CHAIN "NewProc.Bas"
```

مثال٢

```
SELECT CASE Choice$
     CASE "E"
          CHAIN "EditProc.Bas"
     CASE "T"
          CHAIN "Train.Bas"
END SELECT
```

عملية تقليدية

توضح العملية التالية استخدام عبارة CHAIN. ابدأ بتحميل بيسك السريع.

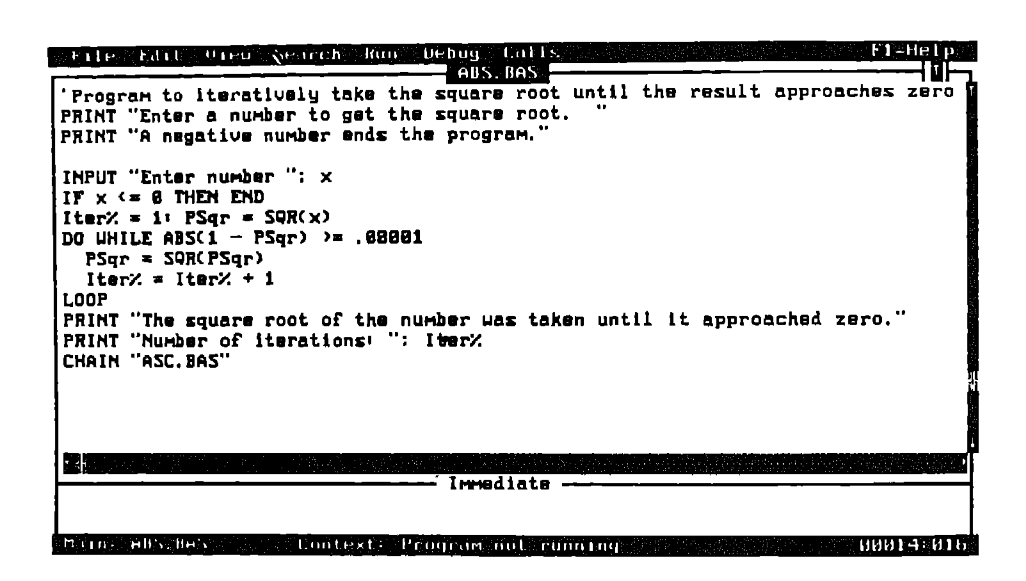
١ - حمل برنامج ABS.BAS وأضف السطر التالي في نهاية البرنامج :

```
CHAIN "ASC.BAS"
```

ملاحظة

إذا لم يشمل متغير PATH من DOS عندك دليل المصدر لبيسك السريع أو إذا لم تكن تعمل ببيسك السريع من دليل المصدر فيجب أن تقدم اسم المسار كجزء من اسم الملف مثلما يلي:

"\basic\qb.4\illqb.arc\Asc.Bas."



```
'Program to iteratively take the square root until the result approaches zero
PRINT "Enter a number to get the square root.  "
PRINT "A negative number ends the program."

INPUT "Enter number ": x
IF x <= 0 THEN END
Iter% = 1: PSqr = SQR(x)
DO WHILE ABS(1 - PSqr) >= .00001
  PSqr = SQR(PSqr)
  Iter% = Iter% + 1
LOOP
PRINT "The square root of the number was taken until it approached zero."
PRINT "Number of iterations: ": Iter%
CHAIN "ASC.BAS"
```

٢ - نفذ البرنامج. بعد تنفيذ بيسك السريع لبرنامج ABS.BAS يظهر ملقن يسألك إذا ما كنت تريد أن تحفظ الملفات ام لا. اكتب N ولاحظ أن البرنامج ينفذ الملف ASC.BAS.

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الثاني والعشرين للاستمرار في تسلسل التعلم.

الدرس الخامس عشر

دالـــة CHR$

الوصف

تعيد دالة CHR$ الرمز الذى تتفق قيمته فى ASCII مع المؤشر الذى يمرر. وتكوينها هو مايلى:

CHR$ (تعبير عددى)

وحيث إن الرمز الذى يمرر يكون من ضمن مجموعة رموز ASCII فدائماً ما يقع التعبير العددى للدالة بين 0 و 255.

التطبيقات

فى البرنامج العينة الموجود فى الدرس الثالث حدث استخدام لهذه الدالة فى طباعة رموز شبه مرسومة من مجموعة ASCII المتسعة أثناء رسم المستطيل. وبعض الاستخدامات الأخرى تحدث لإنتاج سلاسل مكونة من رموز غير الرموز المعتاد استخدامها فى الطباعة. وفيما يلى بعض الأمثلة.

```
OddStr$ = CHR$(13)+CHR$(10)
```

يصف هذا المثال عودة العربة وتسلسل تغذية السطر وعادة ما يستخدم كنهاية لعلامة سجل.

```
OddStr$ = CHR$(34)+
          "Embedded quotes! Otherwise not allowed!"+CHR$(34)
```

يبين هذا المثال كيفية اضافة علامتى تنصيص مزدوجتين إلى سلسلة.

عملية تقليدية

لرؤية الرموز المختلفة الموجودة فى مجموعة ASCII وتوضيح دالة CHR$ فى بيسك السريع استمر كما يلى. ابدأ بتحميل بيسك السريع.

١ – اضغط على F6 للقفز إلى النافذة الفورية، ويمكنك من هذا المكان أن تكتب سطراً من الشفرة وتنفذه على الفور، وتستطيع أن تحصل على معظم المعلومات عن النافذة الفورية من الملحق B.

٢ – اكتب ما يلى :

```
FOR i=1 TO 255 :PRINT i "=" CHR$(i) ; :NEXT
```

٣ – اضغط على مفتاح الادخال لتنفيذ ذلك.

لاحظ أن بعض الرموز التى سبق طباعتها مبكراً أثناء تنفيذ البرنامج أظهرت أشياء غريبة على الشاشة، وهذا لأن هذه الرموز غير قابلة للطباعة. والرموز الوحيدة التى تترك على الشاشة هى التى تبدأ من CHR$ (13) وما بعدها وذلك لأن CHR$ (12) يخلى الشاشة وتطبع الرموز الأخرى منذ بداية الشاشة. يحتوى الملحق C على قائمة كاملة بكل الرموز الموجودة فى مجموعة ASCII.

```
13 =
14 =♫ 15 =☼ 16 =► 17 31 =8 =↕ 19 =‼ 20 =¶ 21 =§ 22 =▬ 23 =↨ 24 =↑ 25 =↓ 26 =→
27 =← 28 =  29  30 =       32 =  33 =! 34 =" 35 =# 36 =$ 37 =% 38 =& 39 =' 40 =(
41 =) 42 =* 43 =+ 44 =, 45 =- 46 =. 47 =/ 48 =0 49 =1 50 =2 51 =3 52 =4 53 =5
54 =6 55 =7 56 =8 57 =9 58 =: 59 =; 60 =< 61 == 62 => 63 =? 64 =@ 65 =A 66 =B
67 =C 68 =D 69 =E 70 =F 71 =G 72 =H 73 =I 74 =J 75 =K 76 =L 77 =M 78 =N 79 =O
80 =P 81 =Q 82 =R 83 =S 84 =T 85 =U 86 =V 87 =W 88 =X 89 =Y 90 =Z 91 =[ 92 =\
93 =] 94 =^ 95 =_ 96 =` 97 =a 98 =b 99 =c 100 =d 101 =e 102 =f 103 =g 104 =h
105 =i 106 =j 107 =k 108 =l 109 =m 110 =n 111 =o 112 =p 113 =q 114 =r 115 =s
116 =t 117 =u 118 =v 119 =w 120 =x 121 =y 122 =z 123 ={ 124 =| 125 =} 126 =~
127 =⌂ 128 =Ç 129 =ü 130 =é 131 =â 132 =ä 133 =à 134 =å 135 =ç 136 =ê 137 =ë
138 =è 139 =ï 140 =î 141 =ì 142 =Ä 143 =Å 144 =É 145 =æ 146 =Æ 147 =ô 148 =ö
149 =ò 150 =û 151 =ù 152 =ÿ 153 =Ö 154 =Ü 155 =¢ 156 =£ 157 =¥ 158 =₧ 159 =ƒ
160 =á 161 =í 162 =ó 163 =ú 164 =ñ 165 =Ñ 166 =ª 167 =º 168 =¿ 169 =⌐ 170 =¬
171 =½ 172 =¼ 173 =¡ 174 =« 175 =» 176 =░ 177 =▒ 178 =▓ 179 =│ 180 =┤ 181 =╡
182 =╢ 183 =╖ 184 =╕ 185 =╣ 186 =║ 187 =╗ 188 =╝ 189 =╜ 190 =╛ 191 =┐ 192 =└
193 =┴ 194 =┬ 195 =├ 196 =─ 197 =┼ 198 =╞ 199 =╟ 200 =╚ 201 =╔ 202 =╩ 203 =╦
204 =╠ 205 =═ 206 =╬ 207 =╧ 208 =╨ 209 =╤ 210 =╥ 211 =╙ 212 =╘ 213 =╒ 214 =╓
215 =╫ 216 =╪ 217 =┘ 218 =┌ 219 =█ 220 =▄ 221 =▌ 222 =▐ 223 =▀ 224 =α 225 =ß
226 =Γ 227 =π 228 =Σ 229 =σ 230 =µ 231 =τ 232 =Φ 233 =Θ 234 =Ω 235 =δ 236 =∞
237 =φ 238 =ε 239 =∩ 240 =≡ 241 =± 242 =≥ 243 =≤ 244 =⌠ 245 =⌡ 246 =÷ 247 =≈
248 =° 249 =∙ 250 =· 251 =√ 252 =ⁿ 253 =² 254 =■ 255 =

Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى النافذة الفورية.

٤ – اضغط على F6 لترك النافذة الفورية والعودة إلى نافذة الرؤية.

٥ – انتقل إلى الدرس المائة والثالث والأربعين للاستمرار فى تسلسل التعلم.

الدرس السادس عشر

عبارة CIRCLE

الوصف

ترسم عبارة CIRCLE دائرة على الشاشة وذلك طبقاً لقيمة مركز ونصف قطر معينين، وتكوينها هو كما يلي :

```
CIRCLE STEP (x,y),radius,color,start,end,aspect
```

يحدد جزء STEP أن احداثيات x و y نسبية للوضع الحالي وهو جزء اختياري، وجزء (x, y) هي احداثيات الشاشة لمركز الدائرة، وجزء radius هو نصف القطر المستخدم في رسم الدائرة، وجزء start وجزء end يستخدمان في رسم جزء من دائرة أو رسم أقواس، وتقع قيم start و end في مدى يتراوح من -2π إلى 2π (حيث π= 3.141593) وقيمهما التقليدية هي 0 لجزء start و 2π لجزء end، وعندما تكون قيمة أي منهما سالبة فيرسم القوس مع اعتبار أن القيمة موجبة، وجزء aspect هو نسبة y دائرياً إلى x دائرياً وتحسب قيمته التقديرية كما يلي :

```
4 * (y pixels/x pixels) / 3
```

حيث نقاط الرسم الرأسية y pixels ونقاط الرسم الأفقية x pixels هما قيمة الثبات للشاشة ويعتمدان على حالة الشاشة، وعندما يكون aspect أقل من 1 فان radius يكون نصف قطر x أما عندما يكون أكبر من 1 فيكون radius هو نصف قطر y.

التطبيقات

عبارة CIRCLE هي وسيلة أخرى متعددة الجوانب في تطبيقات الرسومات، وتقوم العبارة برسم الدائرة كلها أو جزءاً منها أو ترسم قطعاً ناقصا أو قطاعاً منه وذلك طبقاً للمؤشرات، ومقدرة استخدام إحداثيات مطلقة أو نسبية (باستخدام STEP) هي من المميزات الاضافية، وبعد رسم الدائرة تشير عبارة CIRCLE إلى مركزها أي إنك إذا رسمت دائرتين مستخدماً نفس المؤشرات فإنهما يرسمان في نفس المكان إلا إذا حددت احداثيات نسبية، وفيما يلي أمثلة لعبارة CIRCLE :

CIRCLE

```
CIRCLE (180,200),50
CIRCLE (200,100),100,,1,2
CIRCLE (50,200),50,,.05,-.1
```

عملية تقليدية

توضح هذه العملية استخدام عبارة CIRCLE فى صورة مبسطة. استمر إذا ما كان لديك نظام شاشة رسومات ملونة تدعم ذلك فقط. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This program demonstrates the CIRCLE statement.

SCREEN 1: COLOR 1, 3
CLS
FOR cnt = 1 TO 10
  CIRCLE (65, 65), 5 * cnt, , , , 2

NEXT

FOR cnt = 1 TO 10
  CIRCLE (200, 100), 5 * cnt

NEXT

                         Immediate

Main: <Untitled>      Context: Program not running        00013:005
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة CIRCLE فى البرنامج. لاحظ كذلك استخدام مؤشر aspect وكيفية تأثيره على المخرجات.

٣ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ – انتقل إلى الدرس المائة وستة للاستمرار فى تسلسل التعلم.

الدرس السابع عشر

عبارة CLEAR

الوصف

تغلق عبارة CLEAR كل الملفات المفتوحة وتضع قيماً ابتدائية لمتغير البرنامج وتخلى الرصة وتغير حجمها (اختيارياً). وتكوينها هو كما يلى :

```
CLEAR ,,stack
```

يلزم وجود الفاصلتين للتوافقية مع صيغ بيسك القديمة ويعرف جزء stack موقع الرصة الاضافى اللازم بعد الرصة الموجودة حالياً. وايجازاً فعبارة CLEAR تؤدى الأنشطة التالية :

- تغلق كل الملفات المفتوحة وتزيل مكان الذاكرة الاحتياطية.
- تزيل كل متغيرات COMMON.
- تضع أصفاراً كقيم ابتدائية للمتغيرات العددية وتضع فراغات كقيم ابتدائية لمتغيرات السلاسل.
- تخلى الرصة وتغير (اختيارياً) من حجمها.

التطبيقات

يمكن أن تستخدم عبارة CLEAR فى انتاج مكان أكبر للرصة للبرامج التى لها برامج فرعية متداخلة بعمق أو التى لها اجراءات اعادة ذاتية عديدة. وفيما يلى أمثلة لاستخدام عبارة CLEAR.

مثال١

```
CLEAR ,,
```

مثال٢

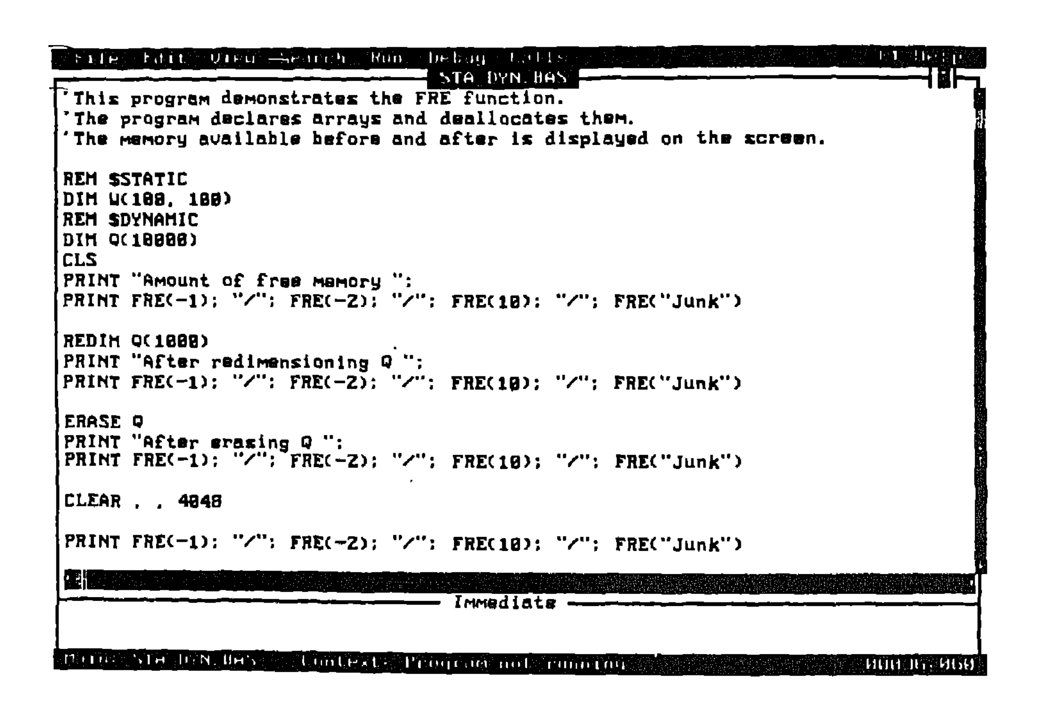
```
CLEAR ,,4048
```

يزيل المثال الأول المتغيرات ويغلق الملفات فقط، أما المثال الثانى فيؤدى هذه الأنشطة بالاضافة إلى أنه يزيد من حجم الرصة بمقدار 4048 بايت. ويجب ألا تستخدم عبارة CLEAR من داخل برنامج فرعى وذلك بسبب اخلاء الرصة وفقدان عنوان العودة إلى عبارة النداء.

عملية تقليدية

هذه العملية تستخدم برنامجاً تم اعداده فى الدرس المائة والسابع والثلاثين وتم تنقيحه فى الدرس الخمسين. وهذا البرنامج معدل هنا لتوضيح استخدام عبارة CLEAR. ابدأ بتحميل بيسك السريع.

١ - حمل البرنامج STA_DYN.BAS وأضف آخر عبارتين كما هو موضح فيما يلى :



```
'This program demonstrates the FRE function.
'The program declares arrays and deallocates them.
'The memory available before and after is displayed on the screen.

REM $STATIC
DIM W(100, 100)
REM $DYNAMIC
DIM Q(10000)
CLS
PRINT "Amount of free memory ":
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

REDIM Q(1000)
PRINT "After redimensioning Q ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

ERASE Q
PRINT "After erasing Q ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

CLEAR , , 4048

PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")
```

٢ - نفذ البرنامج. وتكون المخرجات كما يلى. لاحظ ثالث قيمة فى آخر سطرين والتى تعكس الانكماش فى موقع الرصة.

```
Amount of free memory  262240 / 1162 / 49808 / 49808
After redimensioning Q  298240 / 1162 / 49808 / 49808
After erasing Q  302256 / 1162 / 49808 / 49808
 340272 / 3962 / 47008 / 47008





Press any key to continue
```

٣ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ – انتقل إلى الدرس الثمانين للاستمرار في تسلسل التعلم.

الدرس الثامن عشر

عبارة CLOSE

الوصف

تغلق عبارة CLOSE المدخلات للف أو المخرجات إلى ملف أو إلى أحد الوحدات، وتكوينها هو كما يلى :

```
CLOSE #filenum
```

جزء filenum هو رقم الملف المحدد فى عبارة OPEN، ويمكن للعبارة أن تغلق (اختيارياً) ملفات عديدة مرة واحدة عن طريق كتابة أرقام ملفات اضافية مفصولة عن بعضها البعض بواسطة فواصل.

عندما تستخدم هذه العبارة بدون رقم للملف فإنها تغلق كل الملفات وكل الوحدات. والملف المصاحب لرقم الملف يتوقف عن كونه متاحاً بعد تنفيذ عبارة CLOSE المستخدمة لهذا الرقم كمؤشر لها. وعند ذلك يصبح هذا الرقم متاحاً لإعادة التحديد لأى ملف آخر. وعند استخدام عبارة CLOSE مع ملف تتابعى أو مع وحدة تتابعية فتكتب العبارة آخر ذاكرة احتياطية وتجعل موقع الذاكرة الاحتياطية متاحاً دائماً للبرنامج.

نصيحة : هناك عبارات أخرى لإغلاق كل الملفات ذاتياً وهى CLEAR و END و RESET و RUN و SYSTEM.

التطبيقات

هذه العبارة ضرورية فى تشغيل الملفات وذلك لضمان اتمام العمل بالترتيب المطلوب وسلامة البيانات المخزنة والمستعادة. وفيما يلى أمثلة لهذه العبارة :

مثال١

```
OPEN "Sample.Dat" FOR RANDOM AS #1
..
CLOSE #1
```

مثال٢

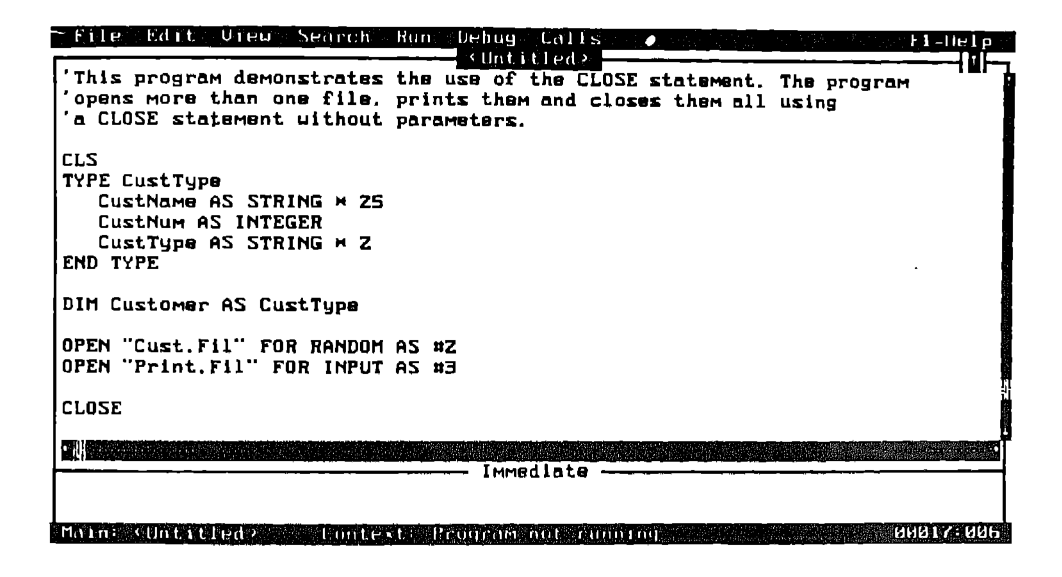
```
OPEN "File.001" FOR INPUT AS #1
OPEN "File.002" FOR OUTPUT AS #2
OPEN "File.003" FOR RANDOM AS #3
OPEN "COM2:1200,7,E,1,ASC" AS #11
..
CLOSE
```

يوضح المثال السابق استخدام هذه العبارة بدون مؤشرات وذلك فى اغلاق كل الملفات والوحدات المفتوحة.

عملية تقليدية

توضح العملية استخدام هذه العبارة. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
'This program demonstrates the use of the CLOSE statement. The program
'opens more than one file, prints them and closes them all using
'a CLOSE statement without parameters.

CLS
TYPE CustType
   CustName AS STRING * 25
   CustNum AS INTEGER
   CustType AS STRING * 2
END TYPE

DIM Customer AS CustType

OPEN "Cust.Fil" FOR RANDOM AS #2
OPEN "Print.Fil" FOR INPUT AS #3

CLOSE
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة CLOSE فى البرنامج.

٣ - ارجع إلى البرنامج ثم اختر New ولا تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الثامن والثمانين للاستمرار فى تسلسل التعلم.

الدرس التاسع عشر

عبارة CLS

الوصف

تخلى هذه العبارة الشاشة وتترك نقطة البداية فى الركن العلوى الأيسر للشاشة. وتكوينها هو كما يلى :

```
CLS 0|1|2
```

الجزء CLS هو إحدى كلمات البيسك المحجوزة، ويمكن أن يتبعه اختيارياً أى من القيم التالية مفصولة عن بعضها البعض بواسطة |. وعندما يستخدم بدون مؤشرات فانه يخلى بوابة المشاهدة النشطة فى الحالة النصية أو فى حالة الرسومات، وفى الحالة النصية يقوم بإعادة نشاط عرضُ مفتاح الوظيفة الموجود فى قاعدة الشاشة إذا ما كان فى الوضع المفتوح. الجزء 0 هو مؤشر اختيارى، ويقوم باخلاء الشاشة من كل النصوص والرسومات. الجزء 1 هو مؤشر اختيارى يقوم باخلاء بوابة مشاهدة الرسومات فقط إذا ما نفذت عبارة VIEW. وإلا فإنه يخلى الشاشة كلها. أما الجزء 2 فهو مؤشر اختيارى يخلى بوابة مشاهدة النصوص فقط مع ترك السطر الموجود فى القاعدة دون أى تغيير.

التطبيقات

تقدم عبارة CLS طريقة فعالة لإخلاء الشاشة وتستخدم كلما دعت الحاجة إلى شاشة جديدة أثناء تنفيذ البرنامج. ويكون هذا مفيداً قبل عرض مخرجات البرنامج أو أخذ مدخلات له. وفيما يلى أمثلة لذلك :

مثال١

```
CLS: INPUT "Enter filename ", F$
```

مثال٢

```
CLS
LOCATE 23,1: PRINT "E)dit / B)rowse / P)rint / Q)uit "
Choice$ = INKEY$
```

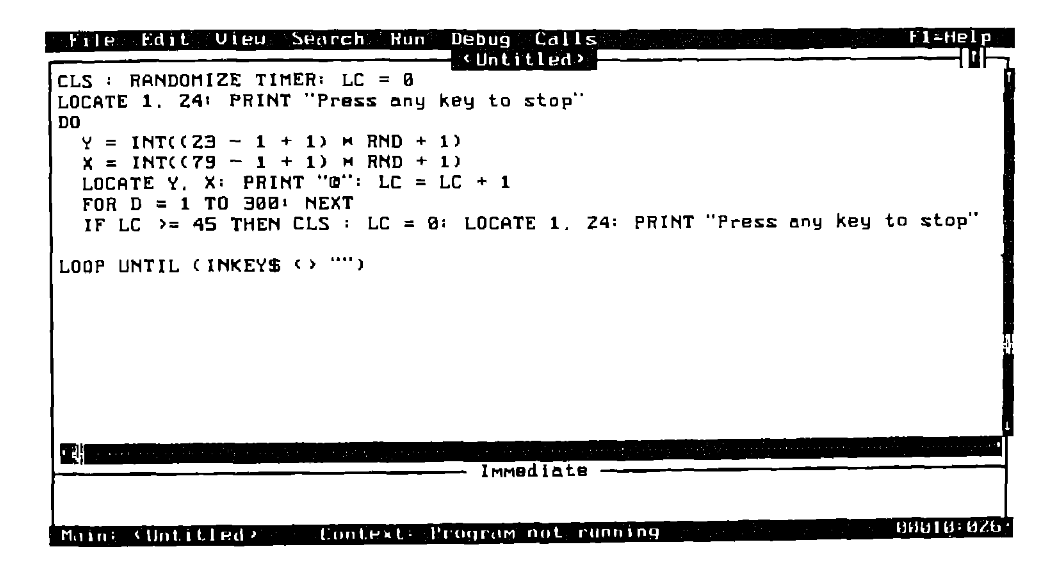
مثال٢

```
GOSUB UpdateRecords
CLS
GOSUB DisplayReports
```

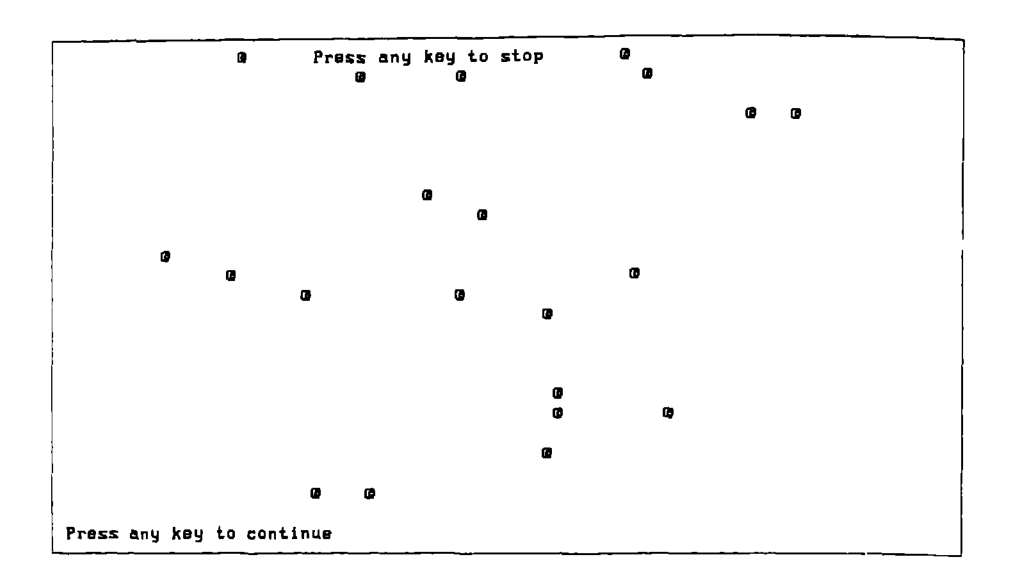
عملية تقليدية

تقوم فى هذه العملية بادخال وتنفيذ برنامج يوضح استخدام هذه العبارة. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - اضغط على Shift-F5 للتنفيذ. لاحظ استخدام عبارة CLS فى اخلاء الشاشة بعد طباعة 45 رمزاً عليها. اضغط على أى مفتاح لإيقاف البرنامج.



٣ – اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ – اضغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٥ – انتقل إلى الدرس الخامس والخمسين للاستمرار فى تسلسل التعلم.

الدرس العشرون

عبارة COLOR

الوصف

تحدد هذه العبارة ألوان العرض، وتكوينها هو كما يلى :

```
COLOR foreground, background, border
COLOR background, palette
COLOR foreground, background
COLOR foreground
```

يستخدم التكوين الأول مع حالة الشاشة 0. وفى حالة الشاشة هذه يمكن اختيار أمامية الشاشة وخلفيتها وحدودها، جزء foreground عبارة عن رقم صحيح يتراوح من 0 إلى 31 ويضع اللون للرموز والأشياء الموجودة على الشاشة، ويعد الرقم 15 تومض الألوان، وجزء background عبارة عن رقم صحيح يتراوح من 0 إلى 7 ويضع اللون لخلفية الشاشة، ومن غير المسموح به أن تومض الألوان فى الخلفية، وجزء border عبارة عن رقم صحيح يتراوح من 0 إلى 15 ويضع حدود الشاشة، ولا يدعم أى من EGA أو VGA أو CGA حدود الشاشة.

ويستخدم التكوين الثانى مع حالة الشاشة 1 والأجزاء عبارة عن أرقام صحيحة تقع فى المدى الذى سبق ذكره، وفى هذه الحالة لا يمكن اختيار لون للأمامية إلا أنه يمكن اختيار لون للخلفية يتراوح من 0 إلى 7، وجزء المجموعة palette عبارة عن رقم فردى أو زوجى تتراوح قيمته من 0 إلى 255 ويختار مجموعة الألوان التى تعرض مع رقم معين للون.

ويستخدم التكوين الثالث فى حالات الشاشة من 7 إلى 10 وجزئى أمامية والخلفية يكونان تعبيرين صحيحين تتراوح قيمتهما فى المدى الذى سبق ذكره، وفى هذه الحالة لا يمكنك أن تحدد حدود الشاشة، وأمامية الشاشة تكون رقم الخاصية أما الخلفية فتكون رقماً لأحد الألوان.

ويستخدم التكوين الرابع فى حالات الشاشة 12 و 13، وتكون الأمامية عبارة عن تعبير صحيح كما سبق ذكره، وتحدد الخاصية لأمامية الرسومات.

التطبيقات

تستخدم عبارة COLOR مع عبارة SCREEN فى تحديد الألوان المستخدمة أثناء ظهور المخرجات على الشاشة. والاختيار المناسب لألوان الشاشة لإحدى حالاتها يكونَ مهماً إذا ماكان البرنامج متداخلاً أو مستخدماً لرسومات وكانت عروض الشاشة مهمة فى أحد التطبيقات. وفيما يلى أمثلة لهذه العبارة.

مثال١

```
SCREEN 1
COLOR 1,2
..
CIRCLE (10,20),40
```

مثال٢

```
COLOR 2,1,1
LINE STEP(100,150)-(50,50)
```

عملية تقليدية

هذه العملية توضح عبارة COLOR فى صورة مبسطة، وذلك مع رسومات تظهر على شاشة أحادية اللون. حمل بيسك السريع أولاً.

١ - اختر Open وحمل البرنامج الذى تم اعداده فى الدرس المائة وتسعة وهو PRINT.BAS.

٢ - نفذ البرنامج. لاحظ المخرجات الموجودة على الشاشة واستخدام عبارة COLOR فى البرنامج.

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الخامس والعشرين للاستمرار فى تسلسل التعلم.

الدرس الحادى والعشـرون

دالـة COMMAND$

الوصف

تعيد دالة COMMAND$ مؤشرات سطر الأمر التى تقدم عند استدعاء البرنامج. وتكوينها هو ما يلى :

COMMAND$

تعيد الدالة سلسلة وهى السلسلة التى سبق طباعتها بعد اسم البرنامج عندما تم استدعاء البرنامج. وتحذف الفراغات الزائدة فى السلسلة وتتحول حروفها إلى الحروف الكبيرة.

وتعيد دالة COMMAND$ مؤشرات سطر الأوامر التى تقدم عند ملقن DOS عندما ينفذ برنامجاً قائماً بذاته. كما أنه من الممكن كذلك توفير مؤشر سطر الأمر عندما ينفذ البرنامج من بيسك السريع عن طريق استخدام خيار cmd/ عند استدعاء بيسك السريع أو عند اختيار COMMAND$ من قائمة Run.

التطبيقات

تستخدم دالة COMMAND$ فى معرفة موقع مؤشرات سطر الأمر. وحيث إن الدالة تعيد سلسلة فقط وليس قائمة بمحتويات السلسلة لسطر الأمر فيكون من وظيفة المبرمج أن يفحص عناصر السلسلة ويعرف المؤشرات. وفيما يلى أمثلة لدالة COMMAND$.

مثال١

```
CmdParm$ = COMMAND$
SELECT CASE CmdParm$
     CASE IS "MONO"
        GOSUB SetMono
     CASE IS "COLOR"
        GOSUB SetColor
END SELECT
```

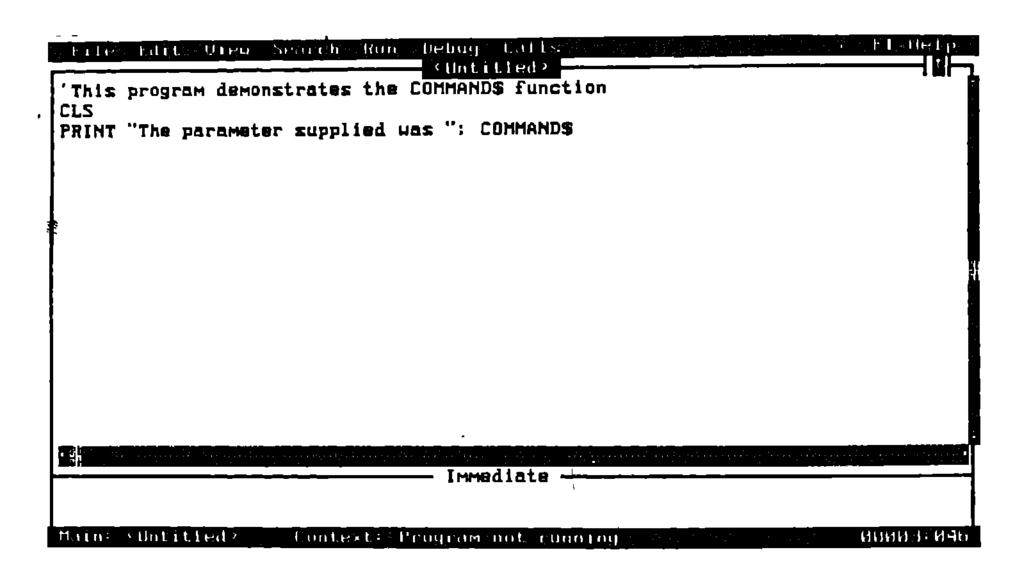
مثال٢

```
DEFSTR A-E
Cparm = COMMAND$
IF Cparm = "" THEN
   GOSUB BusinessAsUsual
ELSE
   GOSUB ParseCommand
END IF
```

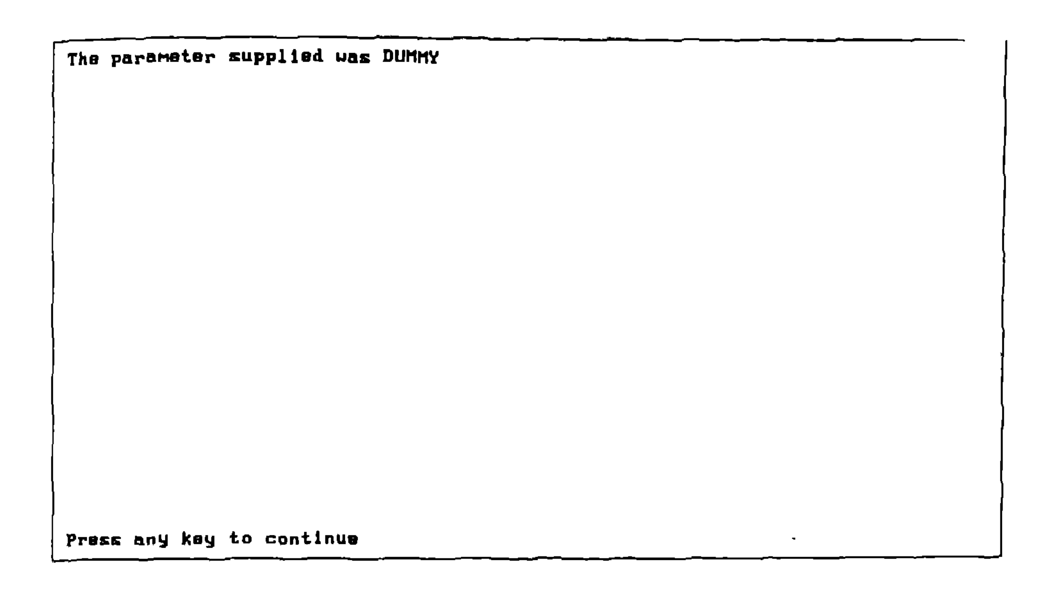
عملية تقليدية

يقبل البرنامج فى هذه العملية مؤشراً واحداً لسطر أمر ويطبع المؤشر المقدم. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



٢ – انتقل إلى قائمة Run واختر COMMAND$ واكتب dummy واضغط على مفتاح الادخال ونفذ البرنامج. تظهر المخرجات على الصورة التالية :



٣ - لاحظ استخدام COMMAND$ فى البرنامج. قم بإخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس السابع والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الثانى والعشرون

عبارة COMMON

الوصف

توضح عبارة المشاركة COMMON المتغيرات على أنها شاملة بحيث يمكن استخدامها بالمشاركة بين البرامج الفرعية والدوال والبرامج المتصلة ببعضها البعض. وتكوين هذه العبارة هو كما يلى :

```
COMMON SHARED /block name/ variable list
```

جزء SHARED اختيارى ويلغى الحاجة إلى وجود عبارة SHARED داخل البرامج الفرعية والدوال لكى تتشارك فى المتغيرات الشاملة. وجزء name/ block هو معرف مجموعة المشاركة، وهو معرف بيسك سريع صحيح، لا يزيد طوله عن 40 خانة. ويعرف اسم المجموعة مجموعة من المتغيرات كمجموعة يمكن اقتسامها أو المشاركة فيها كمجموعة. وعندما يحذف اسم المجموعة فتكون مجموعة المشاركة عبارة عن مجموعة فارغة وإلا فإنها تكون مجموعة مسماة. وعندما يحدث توصيل برامج ببعضها فتستخدم مجموعة مشاركة فارغة نظراً لأن مجموعة المشاركة المسماة لا تستخدم مع البرامج المتصلة ببعضها البعض. ويكون لقائمة المتغيرات الشكل التالى :

```
var1 AS type, var2 AS type,...
```

var1 و var2 هى أسماء متغيرات صحيحة فى بيسك السريع ويحدد نوع AS نوع البيانات التى تنتمى إليها المتغيرات. ويمكن أن يكون النوع من أى نوع من الأنواع البسيطة أو التى يعدها المستفيد. ويمكن توضيح متغيرات المنظومات باستخدام أقواس فارغة. ويجب تحديد ابعاد للمنظومات الاستاتيكية وذلك باستخدام ثوابت عددية فى عبارة DIM وذلك قبل ظهور عبارة المشاركة أما المنظومات الديناميكية فيجب أن توضح فى تسلسل عبارة DIM أو عبارة REDIM قبل استخدامها.

ويجب أن تظهر عبارة المشاركة قبل أى عبارات تنفيذ فى البرنامج وذلك نظراً لأن عبارة المشاركة تحدد مخازن للمتغيرات فى مكان معين فى الذاكرة. ويكون نوع وترتيب المتغيرات الموجودة في مجموعة المشاركة أكثر أهمية من أسماء المتغيرات نفسها. لاحظ المثال التالى :

```
'Program A
COMMON Cnt1, Cnt2 ,Cnt3
Cnt1 = 21: Cnt2 = 0: Cnt3 = 100
..
'Program B
COMMON Cnt2, Cnt3, Cnt1
..
```

قيم Cnt1=100 و Cnt2=21 و Cnt3=0 فى البرنامج وذلك لأن ترتيب ظهورها كان موضحاً فى عبارة المشاركة.

ومن الممكن استخدام مجموعة مشاركة أكبر من مجموعة أخرى إذا ما كانت المشاركة المستخدمة من النوع الفارغ وليس النوع المسمى ولا يمكن اعادة تعريف مجموعة مشاركة لحجم أكبر عندما يقتسمها أحد البرامج مع برنامج فرعى من البرامج الموجودة فى المكتبة.

التطبيقات

عبارة المشاركة هى وسيلة قوية جداً للمشاركة فى استخدام المتغيرات بواسطة برامج فرعية وبرامج متصلة ببعضها البعض. ومع البرامج المتصلة ببعضها البعض تكون مجموعة المشاركة الفارغة هى الطريقة الوحيدة للمشاركة فى استخدام المتغيرات بين البرامج. وتقدم مجموعة المشاركة المسماة طريقة جيدة للمشاركة فى استخدم مجموعة محددة من المتغيرات بواسطة برامج فرعية. وفيما يلى أمثلة لاستخدام عبارة COMMON :

```
'Module 1
COMMON /QGroup/ Q, R, S, T
COMMON /RGroup/ R1, R2, R3
..
'module 2
COMMON /QGroup/ Q, R, S, T
..
```

يوضح المثال السابق كيف يمكن اقتسام استخدام مجموعة مسماة بحيث إنه يمكن الاتصال بمجموعة المتغيرات المحددة. وفى الدرس الثانى، المجموعة المسماة RGroup ليست متاحة.

```
'Module 1
'$INCLUDE: 'ComBlk.001'
..
'module 2
'$INCLUDE: 'ComBlk.001'
..
'ComBlk.001
COMMON SHARED /Pass1/ Lrt, Trt, Yrt
COMMON NewTrt, NewTrt, NewYrt
..
```

يوضح المثال السابق كيف يمكن وضع كل عبارات المشاركة فى ملف شامل مع استخدامها فى الأجزاء المختلفة. ويمكن أن يقلل ذلك بشدة من عبارات المشاركة COMMON غير المتوافقة مع بعضها البعض.

عندما تختلف مجموعات المشاركة فى حجمها فتوضع قيم ابتدائية أصفاراً للمتغيرات العددية وفراغات لمتغيرات السلسلة وذلك لكل المتغيرات الموجودة فى مجموعة المشاركة الأكبر وغير الموجودة كجزء فى مجموعة المشاركة الأصغر.

عملية تقليدية

هذه العملية هى توضيح بسيط لعبارة المشاركة. ويوضح البرنامج أهمية ترتيب ونوع المتغيرات عن أسمائها فى عبارة المشاركة. ويتم إعداد البرنامج كعدة أجزاء منفصلة. ابدأ بتحميل بيسك السريع.

ملاحظة

إذا كانت هناك صعوبة فى هذه العملية فى انتاج الملفات واختيار الخيارات فيجب أن ترجع إلى الملحق B (لاستخدام صناديق الحوار).

١ - اختر خيار Create من قائمة File واكتب COMMON.BAS واضغط على Tab مرتين ثم اضغط على مفتاح الادخال. اكتب أسطر البرنامج التالية :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                               COMMON.BAS
DECLARE SUB Prt ()
'This is module 1
'$INCLUDE: 'ComBlk.001'
Prt1 = 10: Prt2 = 20: Prt3 = 30: Prt4! = .0004
CLS
PRINT "From Module 1:"
PRINT "Prt1 = "; Prt1; "Prt2 = "; Prt2; "Prt3 = "; Prt3; "Prt4! = "; Prt4!
CALL Prt

                               Immediate

Main: COMMON.BAS      Context: Program not running               00008:001
```

٢ - اختر Create من قائمة File وأكتب COMBLK.001. حدد أن نوع الملف هو Include واضغط على مفتاح الادخال ثم اكتب أسطر البرنامج التالية :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                               COMBLK.001
'This file is the COMMON definition include file
COMMON SHARED Prt1, Prt2, Prt3, Prt4!

                               Immediate

Main: COMMON.BAS      Context: Program not running               00003:001
```

٣ - اكتب جزءاً آخر اسمه PRTCOM.BAS واكتب أسطر البرنامج التالية :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         PRTCOM.BAS
'Module 2
COMMON SHARED Prt1, Prt3, Prt4!, Prt2
```

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         PRTCOM.BAS:Prt
SUB Prt
PRINT "From Module 2:"
PRINT "COMMON Prt1, Prt3, Prt4!, Prt2"
PRINT "Prt1 = "; Prt1; "Prt2 = "; Prt2; "Prt3 = "; Prt3; "Prt4! = "; Prt4!

END SUB

                         Immediate
Main: COMMON.BAS      Context: Program not running          00005:001
```

٤ - اختر Save All من قائمة File واحفظ كل الأجزاء التى سبق لك كتابتها.

٥ - اختر Set Main Module من قائمة Run واختر COMMON.BAS كجزء رئيس ونفذه.

تشبه المخرجات ما يلى :

```
From Module 1:
Prt1 =  10 Prt2 =  20 Prt3 =  30 Prt4! =  .0004
From Module 2:
COMMON Prt1, Prt3, Prt4!, Prt2
Prt1 =  10 Prt2 =  .0004 Prt3 =  20 Prt4! =  30




Press any key to continue
```

لاحظ استخدام أمر INCLUDE$ فى الجزء الرئيسى والترتيب المختلف للمتغيرات فى الجزء PRTCOM.BAS ونتائج ذلك. لاحظ كذلك استخدام جزء SHARED فى عبارة المشاركة فى الملف المشمول وجزء PRTCOM. BAS. وفيما يلى بعض الأشياء التى تستحق أن تجربها :

- احذف جزء SHARED ونفذ البرنامج.
- أعد ترتيب المتغيرات ونفذ البرنامج.
- حاول أن تصل الجزء PrtCom. Bas.

٦ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٧ - انتقل إلى الدرس المائة والثانى والعشرين للاستمرار فى تسلسل التعلم.

الدرس الثالث والعشرون

الثابت CONST

الوصف

الثابت Constant عبارة عن قيمة لاتتغير أثناء تنفيذ البرنامج. ولتوضيح أن أحد المعرفات ثابتاً يستخدم فعل CONST. انظر توضيحات الثوابت الموجودة فى البرنامج BOX.BAS المقدم فى الدرس الثالث. الثوابت الموجودة فى هذا الدرس تسمى ثوابت رمزية symbolic constants. أما أنواع الثوابت الأخرى فتسمى ثوابت حرفية literal constants. وفيما يلى بعض الثوابت الرمزية:

```
CONST Pi = 3.1429, FICA = .0751
CONST CarriageRet = 24, EndOfFile = 26
CONST Title$ = "Sir/Madam", Address$ = "Dear " + Title$
CONST State$ = "Texas", City$ = "Dallas"
```

وفيما يلى بعض الثوابت الحرفية :

```
"Thou shalt not."
93000.23
44422
12000
"Todays Appointments: "
```

لاحظ أن الكلمة الرئيسية CONST من بيسك السريع لم تستخدم فى هذه الأمثلة.

ويبين الجدول التالى أنواع الثوابت المختلفة المسموح باستخدامها فى بيسك السريع :

النـــوع	الوصـــف	مثـــال
Integer (Decimal)	قيمة صحيحة تقع فى المدى من 32,768- إلى 32,767 مع وجود اشارة اختيارية تسبقها.	+255,32, -768
Integer (Hexadecimal)	قيمة سادسة عشرية تقع فى المدى من h0 إلى &hFFFF مع وجود &h أو &H تسبقها.	&h10 &HFF
Integer (Octal)	قيمة ثمانية تقع فى المدى من &00 إلى &0177777 مع وجود &0 أو &0 تسبقها.	&010 &0244
Long Integer (Decimal)	مثل الصحيح العشرى (أول نوع فى الجدول) إلا أن المدى يكون فى هذه الحالة من 2,147,483,648- إلى 2,147,483,647+.	&250,000 1,255,768
Long Integer (Hexadecimal)	مثل الثابت السادس عشرى (النوع الثانى فى الجدول) إلا أن المدى يكون فى هذه الحالة من &H0& إلى &FFFFFFFF&. لاحظ ضرورة وجود &.	&H0& &H0DAF&
Long Integer (Octal)	مثل الثابت الثمانى (الحالة الثالثة فى الجدول) إلا أن المدى يكون فى هذه الحالة من &00& إلى &037777777777&. لاحظ ضرورة وجود &.	&O437& &O4447666&
Fixed point	اعداد حقيقية سالبة أو موجبة لها علامات عشرية.	-22.7, 12.44
Floating point	أعداد حقيقية سالبة أو موجبة موجودة فى الصورة الأسية. ويتراوح مداها من 3.37E+38- إلى 3.37E+38.	1.444E+3 256E2 -128.32E-3
Floating point (Double precision)	مثل سابقتها مع استخدام D بدلاً من E فى تحديد الأس. ويتـــراوح المدى من 1.67D+308- إلى 1.67D+308.	1.444D+30
String	رمز واحد أو أكثر موضوع بين علامتى تنصيص مزدوجة. يمكن أن يشمل كل الرموز الموجودة فى ASCII باستثناء رمز علامة التنصيص المزدوجة ورموز اعادة العربة وتغذية السطر. وأقصى عدد للرموز مسموح به فى السلسلة هو 32,767 رمزاً.	"MOVIE" "Costs $.00!"

كل هذه الأنواع سالفة الذكر باستثناء آخر نوع وهو نوع السلسلة هى ثوابت عددية. أما نوع السلسلة فيمثل ثوابت غير عددية أو ثوابت حرفية.

التطبيقات

تستخدم الثوابت لحفظ قيم تستخدم أكثر من مرة واحدة فى البرنامج ولاتتغير هذه القيم أثناء تنفيذ البرنامج. واختيار الثوابت الحرفية بدلاً من الثوابت العددية هو أكثر من اختيارى فتستخدم الثوابت العددية عندما لا يتوقع لها أن تتغير أثناء تنفيذ البرنامج مع استخدام نفس القيمة أكثر من مرة واحدة فى نفس البرنامج. لاحظ الأمثلة المقدمة فى جزء الوصف من هذا الدرس مثال ذلك الثوابت الرمزية Pi و FICA والتى يمكن أن تستخدم فى أماكن عديدة من البرنامج بدون اعادة كتابة القيمة الموجودة فى المتغير. وتكون الثوابت الحرفية خياراً واضحاً عندما يراد استخدام القيمة المحددة مرة واحدة فقط. وأمثلة الثوابت الحرفية الموجودة فى جزء الوصف من هذا الدرس مثالية للحسابات مرة واحدة فقط ولعبارات PRINT كذلك.

عملية تقليدية

فى برنامج BOX. BAS المقدم فى الدرس الثالث يوجد مثال لكيفية استخدام الثوابت. ففى هذا البرنامج تم استخدام الثوابت الرمزية ulx و uly و Lrx و Lry فى حفظ احداثيات الصندوق على الشاشة أثناء تنفيذ البرنامج. ويتغير موضع الصندوق فى هذه العملية مع تغيير مجموعة المتغيرات.

١ - ابدأ ببيسك السريع عن طريق كتابة QB والضغط على مفتاح الادخال ارجع إلى الدرس الثانى والملحق B إذا ما كنت فى حاجة إلى المزيد من المعلومات عن بدء بيسك السريع.

٢ - اضغط على Alt-F لعرض قائمة File.

٣ - اكتب O لتحميل البرنامج. يظهر صندوق حوار فتح الملف. للمزيد من المعلومات عن صندوق الفتح ارجع إلى الملحق B.

٤ - اضغط على Tab للانتقال إلى الدليل.

٥ - باستخدام مفاتيح الأسهم انقل نقطة البداية إلى البرنامج BOX.BAS واضغط على مفتاح الادخال لاختياره.

٦ - غير قيمة Lrx من 40 إلى 50 وغير قيمة Lry من 10 إلى 14.

٧ – اضغط على Shift-F5 لتنفيذ البرنامج. تستخدم الثوابت الرمزية : Horiz و Vert و Tlc و Trc و Brc و Blc فى رسم الرموز للصندوق. فإذا ما غيرتها إلى قيم أخرى فيتم رسم الصندوق برموز أخرى. لتجربة هذه القيم استخدم العملية الموجودة فى الخطوتين 6 و 7.

٨ – اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٩ – اضغط على Alt-F واكتب X للخروج من بيسك السريع. اضغط على Tab وعلى قضيب المسافات لاختيار عدم حفظ التغييرات. تجد نفسك قد عدت الآن إلى ملقن DOS.

١٠ – انتقل إلى الدرس المائة والخامس والخمسين للاستمرار فى تسلسل التعلم.

الدرس الرابع والعشرون

دالـة COS

الوصف

تعيد دالة COS جيب تمام الزاوية المعرفة بالتعبير العددى، وتكوينها هو كما يلى :

COS (تعبير عددى)

وتكون القيمة التى تعود من الدالة بالتقدير الدائرى، فإذا كان التعبير العددى مزدوج الدقة فتكون النتيجة مزدوجة الدقة كذلك وإلا فإنها تكون فردية الدقة. ويمكن أن يكون التعبير العددى من أى نوع من الأنواع المسموح بها فى بيسك السريع.

التطبيقات

تستخدم هذه الدالة عندما يكون مطلوب حساب جيب تمام الزاوية مثل الحسابات المثلثية للرسومات وللألعاب. وفيما يلى أمثلة لدالة COS :

```
T = COS(A) * P
PRINT COS(SIN(Pc))
```

عملية تقليدية

توضح العملية التالية استخدام دالة COS. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
'This program calculates points equally spaced on the
'circumference of a circle.
CLS
PRINT "Calculates points on the circumference of a circle"
INPUT "Enter x,y coordinates of the center of the circle ": x, y
INPUT "Enter the angle of the first point ": Angle
INPUT "Enter number of points to calculate ": Pt
INPUT "Enter the radius of the circle ": radius

Angle = (Angle * 3.14159) / 180
twoPi = 6.28319 / Pt

PRINT "The coordinates :"
FOR cnt = 0 TO Pt - 1
  X1 = x + (radius * COS(Angle + twoPi * cnt))
  Y1 = y + (radius * SIN(Angle + twoPi * cnt))
  PRINT "Point: "; cnt + 1; X1, Y1
NEXT cnt
                              Immediate

Main: <Untitled>     Context: Program not running              00004:019
```

٢ - نفذ البرنامج. اكتب 10 و 10 كمركز و 45 كقيمة للزاوية و 5 كعدد للنقاط و 2 كنصف قطر.
لاحظ استخدام دالة COS في البرنامج. اضغط على أي مفتاح للعودة إلى البرنامج.

```
Calculates points on the circumference of a circle
Enter x,y coordinates of the center of the circle ? 10,10
Enter the angle of the first point ? 45
Enter number of points to calculate ? 5
Enter the radius of the circle ? 2
The coordinates :
Point:  1  11.41421          11.41421
Point:  2  9.092018          11.78201
Point:  3  8.024624          9.687129
Point:  4  9.687135          8.024623
Point:  5  11.78202          9.092024




Press any key to continue
```

٣ - اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والأربعين للاستمرار في تسلسل التعلم.

الدرس الخامس والعشرون

دالـة CSRLIN

الوصف

تعطى هذه الدالة موقع السطر الحالى لنقطة البداية، وتكوينها هو كما يلى :

```
CSRLIN
```

ولا توجد أية مؤشرات لهذه الدالة. والقيمة التى تعيدها الدالة هى السطر الذى توجد عنده نقطة البداية فى الوقت الحالى. ويمكن أن تستخدم هذه الدالة فى أحد التعبيرات.

التطبيقات

هذه الدالة هى أحد الدوال التى تقدم تحكماً فى نقطة البداية أثناء تنفيذ البرنامج، والدوال الأخرى التى تيسر ذلك هى دالة LOCATE ودالة POS. وفيما يلى بعض الأمثلة :

```
LOCATE CSRLIN + 1, 1

SRow = CSRLIN: SCol = POS
LOCATE 24, 1: PRINT "Please choose from above.": BEEP
LOCATE SRow, SCol
```

تستخدم الدالة فى المثال الأول فى تقديم نقطة البداية إلى السطر التالى. ويستخدم المثال الثانى هذه الدالة مع دالة POS فى التذكرة بموقع نقطة البداية مع طباعة رسالة فى أحد المواقع على الشاشة ثم العودة إلى الموقع الأصلى لنقطة البداية.

عملية تقليدية

لقد أعد هذا البرنامج المقدم فى هذه العملية فى الدرس السادس والثمانين. ويوضح البرنامج استخدام برامج فرعية سبق اختبارها اختباراً جيداً لأغراض خاصة. ويقبل البرنامج مدخلات من لوحة المفاتيح طبقاً لمؤشرات محددة. وتحدد المؤشرات السطر والعمود اللذين يجب أن تقبل المدخلات عندهما سلسلة الملقن ومجموعة من الرموز التى تمثل مدخلات صحيحة. وتقوم أنت فى هذه العملية بتعديل البرنامج لطباعة رسالة خطأ فى أحد المواقع على الشاشة والعودة إلى الملقن لادخال مدخلات. ابدأ بتحميل بيسك السريع.

١ - اخــتــر Open وحــمــل البــرنامج الذى ســبق اعــداده فى الدرس الســادس والثــمــانين LRTRIM.BAS.

٢ - عدل الملاحظات ودالة GCH$ إلى ما هو مبين فى السرد التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              CSRLIN.BAS
'This program demonstrates the use of the CSRLIN function.
'The program was created in Module 89; here the function GetChar
'is modified to print a message when the input is incorrect.

DECLARE FUNCTION GCh$ (VCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
DIM VCh AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"

ValidCh = "EDCQ"
Prompt = "Enter selection"
Selection$ = GCh$("EDCQ", 24, 1, Prompt)
```

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              CSRLIN.BAS:GCh
FUNCTION GCh$ (VCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
  DIM Choice AS STRING
  Prompt = LTRIM$(RTRIM$(P))
  Choice = " "
  LOCATE x, y
  PRINT P;

  DO
    Choice = INKEY$
    IF (INSTR(ValidCh, UCASE$(Choice)) = 0) THEN
      CRow = CSRLIN: CCol = POS(0)
      LOCATE 20, 1: Choice = ""
      PRINT "That was not a choice from those available": BEEP
      LOCATE CRow, CCol
    END IF
  LOOP UNTIL (Choice <> "")

  LOCATE x, (y + 2 + LEN(P))
  PRINT Choice;
  GCh$ = Choice
END FUNCTION

------------------------------ Immediate ------------------------------

Main: CSRLIN.BAS      Context: Program not running               00001:001
```

٣ - نفذ البرنامج، اكتب X لتظهر الرسالة، اكتب Q لإنهاء البرنامج لاحظ استخدام هذه الدالة فى حفظ موقع السطر الحالى، وفيما يلى عينة للمخرجات :

```
That was not a choice from those available


E)dit / C)reate / D)elete / Q)uit ?
Enter selection  e
Press any key to continue
```

٤ - ارجع إلى البرنامج واحفظه على أنه برنامج نصي تحت اسم CSRLIN.BAS

٥ - انتقل إلى الدرس المائة وسبعة للاستمرار في تسلسل التعلم.

الدرس السادس والعشرون

دوال CVD و CVI و CVL و CVS

الوصف

دوال CVD و CVI و CVL و CVS هي عكس الدوال MKD$ و MKI$ و MKL$ و MKS$. وتحول هذه الدوال قيم السلاسل الناتجة باستخدام الدوال MKD$ و MKI$ و MKL$ و MKS$ إلى قيم عددية. وتكوينها هو كما يلي :

```
CVD(8 byte string)
CVI(2 byte string)
CVL(4 byte string)
CVS(4 byte string)
```

وتحول دالة CVD سلسلة مكونة من 8 بايت سبق انتاجها بواسطة دالة MKD$ إلى قيمة عددية مزدوجة الدقة.

وتحول دالة CVI سلسلة مكونة من 2 بايت سبق انتاجها بواسطة دالة MKI$ إلى قيمة عددية صحيحة.

وتحول دالة CVL سلسلة مكونة من 4 بايت سبق انتاجها بواسطة دالة MKL$ إلى قيمة عددية صحيحة طويلة.

وتحول دالة CVS سلسلة مكونة من 4 بايت سبق انتاجها بواسطة دالة MKS$ إلى قيمة عددية فردية الدقة.

تستخدم هذه الدوال مع عبارة FIELD فى تحميل بيانات داخل المتغيرات. وتستخدم هذه الدوال فى إبطال عمل التحويلات التى تعد بواسطة الدوال MKD$ و MKI$ و MKL$ و MKS$. وفيما يلى مثال لاستخدامها :

```
OPEN "SalesTx.Dat" FOR RANDOM AS #3 LEN = 42
..
FIELD #3 20 AS ItemName$, 10 AS Qty$, 12 AS SalesTx$
..
GET #3
Quantity = CVI(Qty$): SalesTx! = CVS(SalesTx$)
PRINT ItemName$, Quantity, SalesTx!
```

يوضح المثال السابق كيفية استخدام دالتى CVI و CVS فى تحويل السلاسل إلى قيمها العددية. ويمكن تحقيق نفس التأثير عن طريق انتاج أنواع تكوينات وملفات يقوم المستفيد بتعريفها لهذه الأنواع. ويوضح المثال التالى كيفية استخدام عبارات TYPE و END TYPE فى تحقيق نفس الشىء. لاحظ سهولة وبساطة قراءة الشفرة باستخدام هذه الطريقة.

```
TYPE  SalesTx
     ItemName AS STRING * 20
     Qty      AS STRING * 10
     STax     AS STRING * 12
END TYPE
..
DIM STRec AS SalesTx
OPEN "SalesTx.Dat" FOR RANDOM AS #3 LEN = LEN(STRec)
..
PRINT #3,STRec
```

عملية تقليدية

توضح هذه العملية الدوال CVD و CVI و CVS. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
'This program illustrates the use of CVS, CVD, and CVI functions.
'The program reads in a file created in Module 80, CUSTZ.FIL.

OPEN "CUSTZ.FIL" FOR RANDOM AS #2
FIELD #2, 25 AS CName$, 2 AS CNum$, 8 AS CreditLim$, 4 AS LastInv$
RecCnt = 2
GET #2, 1

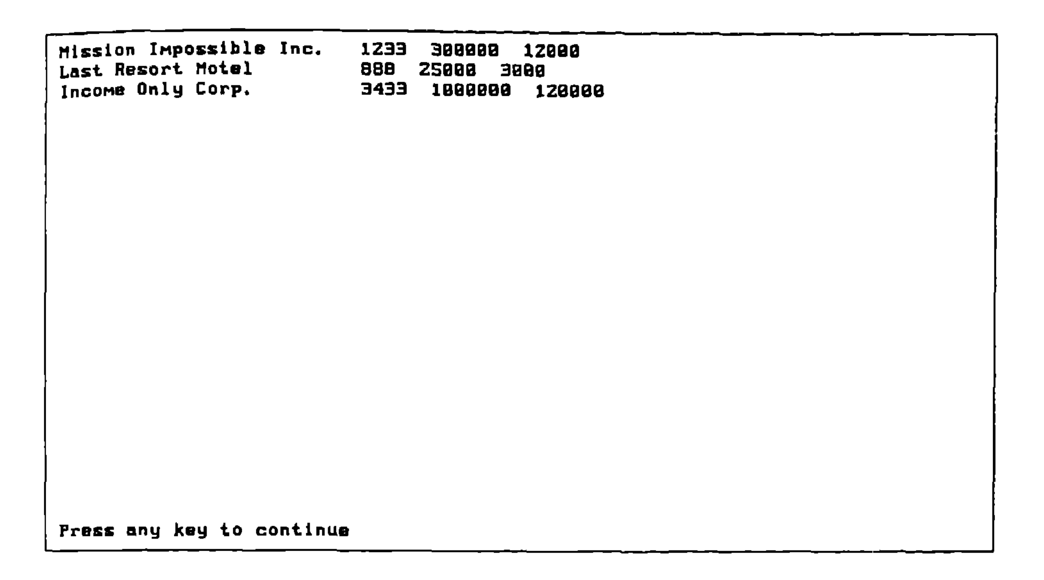
DO WHILE NOT EOF(2)
   PRINT CName$; CVI(CNum$); CVD(CreditLim$); CVS(LastInv$)
   GET #2
LOOP

CLOSE #2

                              Immediate

Main: <Untitled>     Context: Program not running              00010:040
```

٢ - نفذ البرنامج. لاحظ استخدام الدوال سالفة الذكر فى البرنامج.



٣ - ارجع إلى البرنامج واختر New ولا تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس التاسع والثمانين للاستمرار فى تسلسل التعلم.

الدرس السابع والعشرون

عبارة DATA

الوصف

تستخدم عبارة DATA فى دعم عبارة READ بعناصر البيانات. وتكوينها كما يلى :

DATA (قائمة ثوابت)

الجزء	الوصف
DATA	كلمة من كلمات بيسك السريع المحجوزة.
قائمة ثوابت	قائمة بقيم الثوابت مفصولة عن بعضها البعض بواسطة فواصل. ويمكن أن تكون السلاسل عددية أو تكون ثوابت حرفية.

إذا ظهرت ثوابت رمزية (ثوابت معرفة فى عبارة CONST) فى عبارة DATA فتفسر هذه الثوابت كقيمة سلسلة وليست كالقيمة المعرفة فى عبارة CONST وفيما يلى مثال لذلك :

```
CONST MaxList = 200
..
READ Vall,Val2
..
DATA 100,MaxList
```

إذا نفذت عبارة READ فيفسر عنصر البيانات الثانى (MaxList) كثابت سلسلة (ثابت حرفى) وليس كالقيمة العددية 200.

اعتبارات أخرى : يمكن أن تحتوى عبارة DATA على أى عدد من الثوابت يمكن كتابته فى سطر واحد. ويمكن أن تظهر العديد من عبارات DATA فى أسطر متتالية. يمكن أن تحتوى عبارة DATA على أكثر من قيمة واحدة تتطلبها عبارة READ. وتستمر عبارة READ من أخر عنصر بيانات لم يسبق قراءته من القائمة. ويجب أن تحتوى عبارة DATA على عناصر بيانات لايقل عددها عن عدد المتغيرات الموجودة فى عبارة READ. وإذا كانت هناك عناصر بيانات أقل من المتغيرات فيحدث خطأ وقت التشغيل.

التطبيقات

تستخدم عبارة DATA مع عبارة READ فقط. وعلى هذا فيجب أن تظهر مرة واحدة على الأقل فى أى برنامج توجد به إحدى عبارات READ. ويجب أن تتفق كذلك مع متطلبات التكوين واللغة والمنطق. وفيما يلى بعض الأمثلة :

مثال١

```
READ Qty%.Rate#.Invoice$
..
DATA 2300.190.23.QR9989
```

مثال٢

```
TYPE  Employee
   EmpName AS STRING*35
   EmpNum  AS STRING*5
   Dept    AS STRING*20
   EmpRefDetail AS STRING*10
END TYPE
DIM NewEmployee AS Employee
..
READ NewEmployee.EmpName,NewEmployee.EmpNum.NewEmployee.Dept
..
DATA Jon X. Hancock.WS3332,Food&Bev
```

عملية تقليدية

يوضح البرنامج الموجود فى هذه العملية استخدام عبارة DATA. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  ...  ug  Calls                              F1=Help
                         <Untitled>
'This program demonstrates the DATA statement
TYPE Employee
   EmpName AS STRING * 35
   EmpNum  AS STRING * 5
   Dept    AS STRING * 20
   EmpRefDetail AS STRING * 10

END TYPE

DIM NewEmp AS Employee
READ NewEmp.EmpName, NewEmp.EmpNum, NewEmp.Dept
```

```
CLS
PRINT "This is a demonstration of the DATA statement"
PRINT : PRINT
PRINT "Employee Name       : "; NewEmp.EmpName
PRINT "Employee Number     : "; NewEmp.EmpNum
PRINT "Department          : "; NewEmp.Dept
PRINT "Employee Reference : "; NewEmp.EmpRefDetail
DATA Jon X. Hancock, US332, Food&Bev

───────────────── Immediate ─────────────────

Main: <Untitled>      Context: Program not running        00001:037
```

٢ – نفذ البرنامج ولاحظ استخدام عبارة DATA فى البرنامج.

```
This is a demonstration of the DATA statement


Employee Name       : Jon X. Hancock
Employee Number     : US332
Department          : Food&Bev
Employee Reference :










Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس المائة والتاسع والأربعين للاستمرار فى تسلسل التعلم.

الدرس الثامن والعشـرون

دالة وعبارة DATE$

الوصف

يمكن استخدام هذه الكلمة الرئيسية كعبارة أو كدالة وذلك مثل الكلمة الرئيسية TIME$. وكعبارة تضع هذه الكلمة تاريخ الكمبيوتر أما كدالة فإنها تعيد التاريخ الحالى للكمبيوتر. وتكوينها فى كل من الحالتين هو كما يلى :

تكوين العبارة:

DATE$ = تعبير سلسلة

الجـزء	الوصـــف
DATE$	كلمة من كلمات بيسك السريع المحجوزة.
تعبير سلسلة	يمكن لتعبير السلسلة أن يكون متغيراً أو تعبيراً ثابتاً. ويجب أن يكون فى أحد الصور التالية : mm-dd-yy أو mm-dd-yyyy أو mm/dd/yy أو mm/dd/yyyy حيث mm هى الشهر و dd هى اليوم و yy أو yyyy هى السنة.

تكوين الدالة : وتكوين الكلمة كدالة هو كما يلى :

```
DATE$
```

وتعيد هذه الدالة سلسلة فى الصورة mm-dd-yyyy تعبر عن التاريخ الحالى للكمبيوتر.

التطبيقات

تستخدم هذه الكلمة أساساً فى قراءة ووضع تاريخ نظام الكمبيوتر. وهى مفيدة جداً عند كتابة تقارير باستخدام الطابع. كما يمكن أن تستخدم كذلك فى حفظ تتبع تجديدات الملفات عن طريق إضافة سجل آخر إلى الملف مع وجود تاريخ التجديد عليه. كما توجد تطبيقات عديدة أخرى كذلك. وفيما يلى بعض الأمثلة :

أمثلة لعبارة DATE$:

```
DATE$ = "12/31/88"
DATE$ = "01/22/1987"
DATE$ = "02-02-80"
DATE$ = "11-01-79"
```

أمثلة لدالة DATE$:

```
CurrDate$ = DATE$
PRINT DATE$
```

عملية تقليدية

توضح هذه العملية استخدام الكلمة كدالة. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

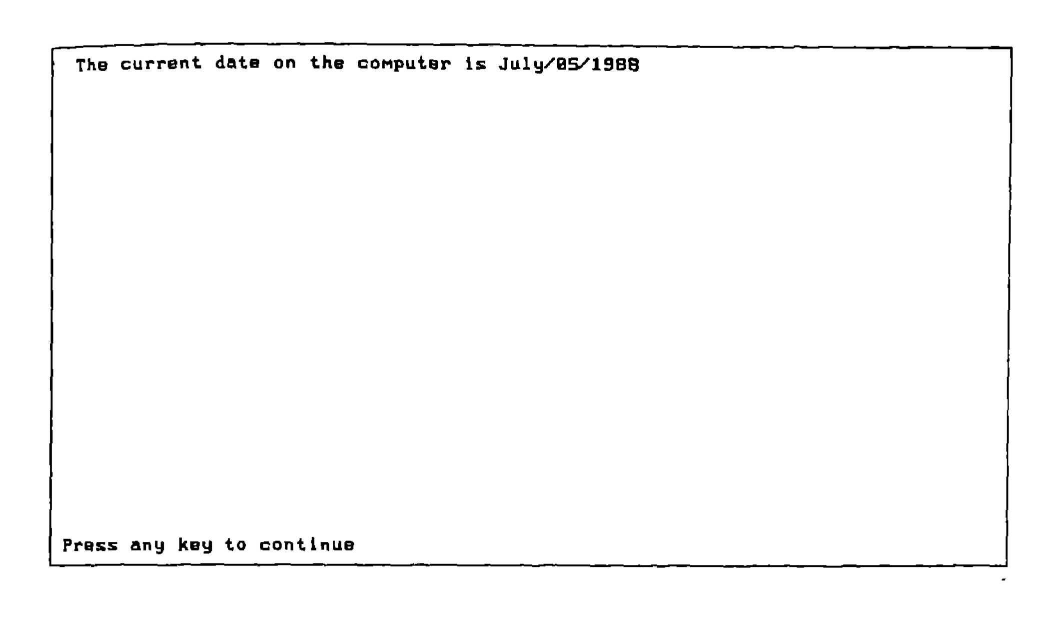
```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              DATE.BAS
'This program demonstrates the DATE$ verb
DIM Month$(12)
FOR m = 1 TO 12: READ Month$(m): NEXT m
CLS
D$ = DATE$
PRINT "The current date on the computer is ": Month$(VAL(D$)): "/";
PRINT MID$(D$, 4, 2): "/": MID$(D$, 7, 4)

DATA January, February, March, April, May, June, July
DATA August, September, October, November, December

                              Immediate

Main: DATE.BAS     Context: Program not running              00001:001
```

٢ – نفذ البرنامج ولاحظ المخرجات واستخدام دالة DATE$ فى البرنامج.



٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F واضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثامن والأربعين للاستمرار فى تسلسل التعلم.

الدرس التاسع وعشرون

عبارة DECLARE

الوصف

تتسبب هذه العبارة فى أن يشير البرنامج الرئيسى إلى برامج بيسك سريع فرعية وإلى اجراءات فرعية ويبدأ عملية التأكد من نوع المؤشر. وتستخدم هذه العبارة للاشارة إلى اجراءات فى بيسك السريع وكذلك اجراءات مكتوبة بكل من لغتى المجمع و C. وفيما يلى تكوين العبارة. (يحدد FUNCTION | SUB استخدام احدى الكلمتين المحجوزتين وليس الاثنتين معاً).

اجراءات بيسك السريع:

```
DECLARE FUNCTION|SUB name (parameterlist)
```

غير اجراءات بيسك السريع :

```
DECLARE FUNCTION|SUB name CDECL ALIAS "Alias name"(parameterlist)
```

اجراءات بيسك السريع : جزء name هو اسم للدالة FUNCTION أو للبرنامج الفرعى الذى يستخدم فى عبارات التحديد واستدعاءات الإجراء. ويكون الاسم معرفاً صحيحاً من معرفات بيسك السريع ويمكن أن يشغل حتى 40 خانة كحد أقصى. وتأخذ قائمة المؤشر param- eter list التكوين التالى :

```
Var1 AS type, var2 AS type,..
```

جزئى var1 و var2 عبارة عن متغيرات من متغيرات بيسك السريع ويعرف جزء AS type نوع البيانات للمتغيرات. ويمكن أن يكون هذا النوع من النوع البسيط أو النوع الذى يعرفه المستفيد. وغير مسموح باستخدام سلاسل ثابتة الطول. وعندما يكون المتغير منظومة فيلى المتغير قوسان فارغان على النحو التالى :

```
var1(), var2(),..
```

ويحدد شكل العبارة ما إذا كان نوع القوائم قد اختبر أم لا. ويوضح الجدول التالى ذلك :

التكوين	كيفية أداء اختبار نوع القائمة
DECLARE SUB name1	لا يسرد أى مؤشرات أساسية. ولا يستخدم أى اختبار للنوع.
DECLARE SUB name ( )	لا يوجد فيه أى مؤشرات. محاولة المرور عند المؤشر لهذا البرنامج الفرعى تتسبب فى حدوث خطأ.
DECLARE SUB name1 (var1 AS INTEGER)	لا يوجد مؤشر فى SUB. ويحدث اختبار لنوع القائمة.

وتستخدم عبارة DECLARE عند استدعاء برنامج فرعى بدون الكلمة المحجوزة CALL أو عند استدعاء برنامج فرعى في جزء آخر. وينتج بيسك السريع هذه العبارات للجزء قبل أن يكتب الجزء فى الملف. ومثل هذه العبارات التى تظهر فى أحد الأجزاء تكون مخصصة لهذا الجزء فقط وتؤثر على محتوياته.

غير اجراءات بيسك السريع : فى هذا التكوين جزء name هو اسم صحيح من اسماء متغيرات بيسك السريع ويمكن أن يشغل حتى 40 خانة كحد أقصى. ويحدد جزء CDECL أن البرنامج الفرعى المستدعى، أو الدالة المستدعاة، هو برنامج فرعى مكتوب بلغة C. واستخدام ذلك يؤثر على كيفية البحث فى المكتبات وفي ملفات التشغيل object files للبرنامج الفرعى. وعند استخدام CDECL يتحول الاسم إلى حروف صغيرة ويدمج رمز الشرطة التى تقع تحت الحرف قبله كما تمرر القائمة كذلك من اليمين إلى اليسار بدلاً من معدل بيسك السريع المعتاد من اليسار إلى اليمين. ويحدد جزء ALIAS أن البرنامج الفرعى له اسم مختلف فى المكتبة أو فى ملفات التشغيل. وجزء "Alias name" هو اسم البرنامج الفرعى الموجود فى المكتبة. وعند استخدام الكلمة المحجوزة ALIAS يستخدم "Alias name" فى البحث فى المكتبة وفى ملفات التشغيل. ولقائمة المؤشرات الشكل التالى :

```
BYVAL|SEG var1 AS type, BYVAL|SEG var2 AS type,...
```

يحدد جزء BYVAL|SEG كيفية تمرير القائمة إلى البرنامج الفرعى. وتستخدم الكلمة المحجوزة BYVAL إذا ما تم تمرير القائمة طبقاً للقيمة. وتستخدم الكلمة المحجوزة SEG إذا ما تم تمرير القائمة كعنوان مجزأ. وأجزاء var1 و var2 و AS type هى نفسها مثلما سبق وصفه فى القسم الخاص بتوضيح اجراءات بيسك السريع. ويتسبب استخدام الكلمة المحجوزة

ANY بدلاً من AS type فى اجتياز اختبار نوع القائمة. وعند استخدام BYVAL فى قائمة المؤشرات يمكنك ألا تستخدم الكلمة المحجوزة ANY.

يعتمد اختبار النوع على شكل عبارة DECLARE. ارجع إلى القسم الخاص بهذه العبارة فى اجراءات بيسك السريع.

تخزين المتغير

يناقش هذا القسم كيفية تخزين بيسك السريع للمتغيرات. وتفيد معرفة ذلك عند استخدام برمجة بلغات مختلطة أو عند استخدام احدى الدوال التالية :

SADD
SETMEM
VARPTR
VARSEG
VARPTR$

يخزن بيسك السريع المتغيرات فى منطقة تسمى DGROUP (جزء البيانات التقليدى للبرنامج) أو كعناوين بعيدة. وعند تنفيذ البرنامج كبرنامج قائم بذاته تخزن كل المتغيرات البسيطة والمنظومات الاستاتيكية STATIC$ والمنظومات الديناميكية DYNAMIC$ لسلاسل المتغيرات فى DGROUP. ويمكن الاشارة إلى ذلك باستخدام عناوين ومشيرات قريبة. أما الأنواع الأخرى للمنظومات الديناميكية فتخزن كعناوين بعيدة. وعند تنفيذ البرنامج فى بيئة بيسك السريع توضع كل المتغيرات البسيطة والمنظومات الاستاتيكية فى عبارة مشاركة وتخزن منظومات سلاسل المتغيرات فى DGROUP وتخزن بقية المنظومات الأخرى كلها كأشياء objects بعيدة.

والأشياء التى يمكن أن تقود إلى متغيرات تنقل فى الذاكرة هى ما يلى :

- اشارة إلى ثابت سلسلة أو تعبير سلسلة.
- استدعاء DEF FN أو FUNCTION.
- استخدام دوال سلاسل أو دوال مرتبطة بالذاكرة.
- اشارة إلى منظومة محدد لها أبعاد ضمنياً.

التطبيقات

هذه العبارة تمثل آلية تحكم لضمان أن كل البرامج الفرعية التى يشار إليها فى برنامج بيسك السريع متاحة وأن المؤشرات التى تمرر تكون متوافقة مع المؤشرات الأساسية. وينتج بيسك السريع هذه العبارات (عبارات DECLARE) فى الحالات التى تكون فيها البرامج الفرعية جزءاً من المقطع، وعندما لا تكون كذلك فيجب أن تكتب عبارة توضيح DECLARE فى ملف $INCLUDE واحد. ومع هذه الطريقة لاتقلق من وجود كل عبارات التوضيح هناك. وعبارات التوضيح هذه ضرورية عند استدعاء البرنامج الفرعى بدون استخدام الكلمة المحجوزة CALL. ويقوم بيسك السريع بإدخال عبارات التوضيح حتى إذا كان المقطع يستخدم كلمة CALL المحجوزة فى عبارة الاستدعاء. وفيما يلى بعض الأمثلة لهذه العبارة :

```
DECLARE FUNCTION MoveRight$(AnyStr$)
DECLARE FUNCTION MoveLeft$
DECLARE FUNCTION CenterText$ ()
```

تشمل عبارة التوضيح الأولى قائمة مؤشرات أساسية وينفذ اختبار النوع. ولا توجد قائمة مؤشرات أساسية فى عبارة التوضيح الثانية ولا ينفذ اختبار النوع. ولا يعنى هذا أن FUNC-TION Moveleft$ ليست لها أية مؤشرات، وإنما لا ينفذ اختبار النوع على القوائم التى تمر إلى الدالة فقط. وتشمل عبارة التوضيح الثالثة قائمة مؤشرات فارغة تتسبب فى ظهور رسالة خطأ إذا ما حدثت محاولة لتمرير أى مؤشرات.

عملية تقليدية

توضح هذه العملية كيفية استخدام عبارة التوضيح فى أحد البرامج. وقد سبق إعداد البرنامج فى الدرس الحادى والسبعين. ابدأ بتحميل بيسك السريع.

١ – اختر Open وحمل البرنامج LUBOUND.BAS.

٢ – لاحظ عبارة التوضيح فى البرنامج. عدل عبارات CALL إلى ماهى عليه فى السرد التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              LUBOUND.BAS
'The following program demonstrates the use of the UBOUND and LBOUND
'statements. The program loads two sets of array values and
'finds the minimum and maximum values in those arrays.

DECLARE SUB FindMinMax (A%(), MinVal, MaxVal)

Max = 15
DIM A%(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A%(Cnt)
  NEXT
  RETURN


Start:
  CLS
  GOSUB LoadArray
  PRINT "First pass"
  FindMinMax A%(), MinVal, MaxVal
  READ Max
  REDIM A%(Max)
  GOSUB LoadArray
  PRINT "Second pass"
  FindMinMax A%(), MinVal, MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,80,67,54,10
DATA 10
DATA 8,89,76,54,23,32,12,4,33,54
SUB FindMinMax (A%(), MinVal, MaxVal)
 MinVal = A%(1): MaxVal = A%(1)
 FOR Cnt = LBOUND(A%) + 1 TO UBOUND(A%)
   IF MinVal > A%(Cnt) THEN
      MinVal = A%(Cnt)
   END IF
   IF MaxVal < A%(Cnt) THEN
      MaxVal = A%(Cnt)
   END IF
 NEXT
 PRINT "Minimum value in array: "; MinVal, "Maximum value in array: "; MaxVal
END SUB

                              Immediate

 Main: LUBOUND.BAS     Context: Program not running                 00001:001
```

٣ - نفذ البرنامج. تظهر شاشة المخرجات على النحو التالى :

```
First pass
Minimum value in array:  1  Maximum value in array:  656
Second pass
Minimum value in array:  4  Maximum value in array:  89




Press any key to continue
```

٤ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٥ – انتقل إلى الدرس المائة والرابع والأربعين للاستمرار فى تسلسل التعلم.

الدرس الثلاثون

عبارة DEF FN

الوصف

تقوم عبارة DEF FN بتعريف احدى الدوال، وتوجد طريقتان لعمل ذلك وتكوين العبارة هو كمايلى :

التكوين الأول (سطر واحد) :

```
DEF FNname (parameter list) = expression
```

التكوين الثانى (أسطر متعددة) :

```
DEF FNname (parameter list)
     ..
     FNname = expression
     ..
END DEF
```

جزء name المصاحب للكلمة المحجوزة FN يعطى للدالة اسمها، و name هو اسم متغير صحيح فى بيسك السريع (يمكن أن يصل طوله حتى 40 خانة كحد أقصى محدداً أى نوع لرمز التوضيح)، وفيما يلى مثال لذلك :

```
DEF FNToken$ (InString$) = MID$(InString$,1,10)
```

جزء parameter list هو قائمة بأسماء متغيرات مع استخدام الفواصل بينها، ويمكن أن يوجد فى قائمة الأسماء توضيح اختيارى لنوع المتغيرات كجزء من القائمة، وفيما يلى مثال لذلك :

```
(TempInC AS SINGLE, TempInF AS SINGLE, Count AS INTEGER)
```

وتحدد القيم لهذه المتغيرات من البيئة المنادية، وتمرر كل المؤشرات طبقاً للقيمة، وهناك طريقتان لتمرير المؤشرات إلى الإجراء أو إلى الدالة وذلك طبقاً للقيمة أو طبقاً للدليل، وكل من الطريقتين مشروحة فى المقاطع التالية.

المرور طبقاً للقيمة : فى هذه الحالة تنسخ محتويات المتغيرات من البيئة المنادية داخل قائمة المؤشرات فى الدالة، وتكون هذه المتغيرات محلية للدالة نفسها ولا تؤثر على ذلك على أى متغير من المتغيرات على المستوى الشامل.

المرور طبقاً للدليل : فى هذه الحالة تتأثر المتغيرات فى البيئة المنادية بأى تعديل يجرى على المتغيرات فى قائمة المؤشرات للدالة.

ويحدث فى التكوينين 1 و 2 حساب قيمة التعبير expression وتحديد النتيجة لاسم الدالة FN name. ويشمل جسم الدالة تعبيراً فى التكوين 1. أما فى التكوين 2 فيكون التعبير جزءاً فقط من التحديد. وهذه هى طريقة اعادة الدالة للنتيجة إلى البيئة المنادية. وعند ترك ذلك، تعيد الدالة صفراً لعبارة DEF FN العددية وفراغاً لعبارة DEF FN الحرفية (السلسلة).

اعتبارات أخرى : فيما يلى قائمة بالأشياء التى يجب تذكرها عن عبارة DEF FN :

- يجب أن تعرف الدالة قبل أن يمكن استخدامها. فإذا ما لم يتحقق ذلك فتظهر رسالة تفيد بأن الدالة غير معرفة.
- لا يمكن أن تتداخل الدوال مع بعضها البعض. فلا يمكن أن تظهر عبارة DEF FN داخل عبارة DEF FN أخرى.
- لا يمكن لدالة معرفة بواسطة عبارة هذا الفصل أن تكون لها اعادة ذاتية، فلا تستطيع الدالة استدعاء نفسها.
- تستخدم عبارة EXIT DEF للخروج بصفة دائمة من DEF FN.
- يجب أن توضح عبارة DEF FN المشار إليها بدليل فى نفس المقطع. فهى محلية بالنسبة إلى المقطع (ملف المصدر).
- كن حذراً عند اعادة اصدار أوامر تعبيرات داخل بيئة وقت تشغيل البيسك لأغراض الكفاءة. يمكن لبيسك السريع أن يعيد اصدار أمر تعبير دون أن يخطر المبرمج بذلك. ويمكن تجنب الآثار الجانبية لهذا عن طريق عزل استدعاء الدالة. وفيما يلى مثال لذلك :

```
DEF FNMult
    InNum = 3
    FnMult = InNum * 10
END DEF
InNum = 1
PRINT FNMult + 23 + InNum
```

وعندما يعاد أمر اصدار تعبير فى عبارة PRINT مثل استدعاء FNMult بعد 23 + InNum تكون النتيجة 54 بدلاً من 56.

- المتغيرات غير الموضحة على أنها استاتيكية داخل عبارة DEF FN وليست جزءاً من قائمة المؤشرات تكون عاملة فى هذا المقطع.

- القيم التى تعود بواسطة عبارة DEF FN تكون من نفس نوع الدالة بغض النظر عن تعبير المصدر. عندما تتحدد قيمة عددية لدالة سلسلة أو العكس عندما تتحدد قيمة سلسلة لدالة عددية تظهر رسالة بعدم مواءمة الدالة.

التطبيقات

تزداد فائدة عبارة DEF FN فى عملية عزل أجزاء من شفرة سبق اختبارها اختباراً جيداً وتنفيذ أنشطة معرفة تعريفاً جيداً بحيث إن البرنامج يكون مرتباً وسهلاً فى قراءته. واعداد مكتبات مصدر بمثل هذه الدوال مفيد للغاية. وفيما يلى بعض الأمثلة :

عبارة DEF FN فى سطر واحد :

```
DEF FNChiS& = ([D - E)^2)/E
DEF FNCirc& = (PiValue * Radius) ^2
DEF FNTruncStr$ = RIGHT$(AnyStr$,10)
```

عبارة DEF FN فى عدة أسطر :

مثال١

```
DEF FNStepUp(InValue)
     FnStepUp = InValue * 2
END DEF
```

مثال٢

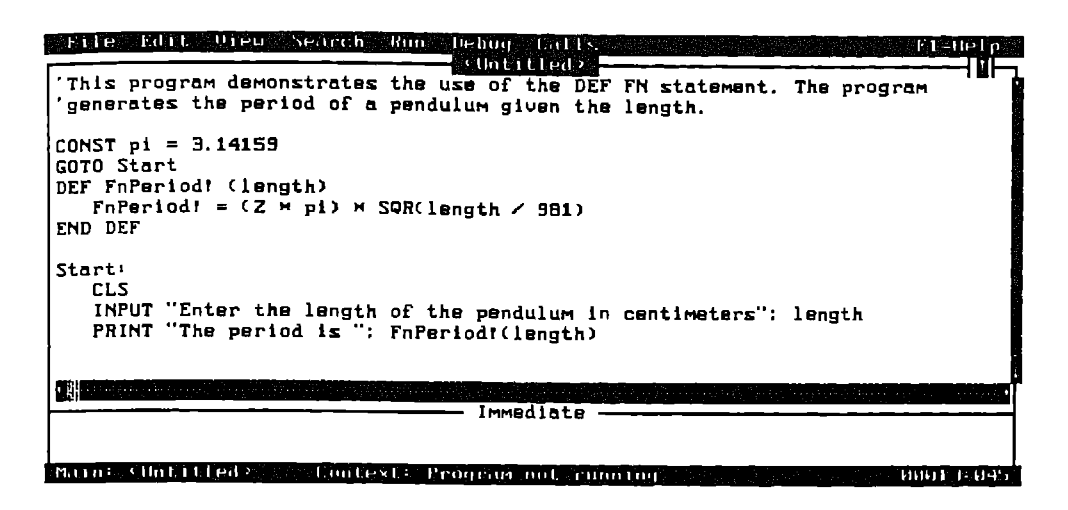
```
DEF FNPayrollUpdate(Name$, Address$, NewPScale)
     ..
     ..
     FNPayrollUpdate = 0
END DEF
```

يعيد هذا المثال صفراً عندما يتم اجراء العملية بنجاح.

عملية تقليدية

هذه العملية توضح استخدام عبارة DEF FN. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



```
'This program demonstrates the use of the DEF FN statement. The program
'generates the period of a pendulum given the length.

CONST pi = 3.14159
GOTO Start
DEF FnPeriod! (length)
   FnPeriod! = (2 * pi) * SQR(length / 981)
END DEF

Start:
   CLS
   INPUT "Enter the length of the pendulum in centimeters": length
   PRINT "The period is "; FnPeriod!(length)
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة DEF FN فى البرنامج.

```
Enter the length of the pendulum in centimeters? 20
The period is  .8971395













Press any key to continue
```

٣ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ – انتقل إلى الدرس التاسع والعشرين للاستمرار فى تسلسل التعلم.

الدرس الحادى والثلاثون

عبارة DEF SEG

الوصف

ذاكرة الكمبيوتر مقسمة إلى قطاعات كل منها يتكون من 64 كيلو بايت. ويتكون عنوان الذاكرة من جزءين هما القطاع segment والفرع offset. ويقوم الكمبيوتر بايجاد البيانات الموجودة فى الذاكرة عن طريق الاتصال بالقطاع أولا ثم البحث عن البيانات داخل القطاع فى أحد أفرع هذا القطاع. وعندما تنفذ البرامج فإنها تحتل قطاعات معينة وتنتج عناوين بيانات تبدأ بقطاعات بدايتها.

وعبارة DEF SEG تضع عنوان القطاع الحالى كقيمة لقطاع جديد. وتكوينها هو كما يلى:

```
DEF SEG = address
```

جزء address هو تعبير عددى تقع قيمته فى المدى من 0 إلى 65,535. ويوضع هذا العنوان كقطاع ذاكرة جديد لكل من PEEK و POKE و BLOAD و BSAVE و CALL ABSOLUTE. وعندما يقع العنوان خارج المدى المحدد تظهر رسالة تفيد بأن استدعاء الدالة غير مسموح به. وعندما يحذف جزء العنوان address فيحدث تحديد لقطاع تقليدى كقطاع بيسك السريع للبيانات.

التطبيقات

تنبيه

هذه العبارة عبارة جديدة ويجب أن تستخدم مع الحذر الشديد. حيث إنه إذا ما تم اختيار قطاع غير مناسب ونفذت سلسلة من PEEK و POKE أو تمت أى محاولة للاتصال المباشر للذاكرة فتكون النتائج وخيمة.

والاستخدام الآمن لهذه العبارة عادة ما يكون مع عبارات BLOAD و BSAVE. وفيمايلى بعض الأمثلة لهذه العبارة :

مثال١

```
DEF SEG = 0
PEEK(..)
POKE ...
```

مثال٢

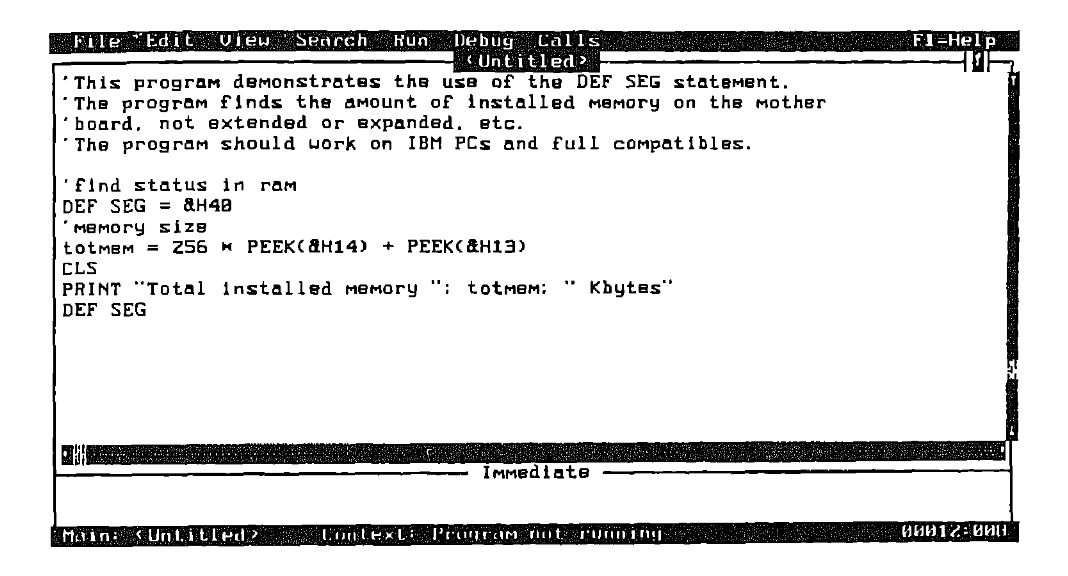
```
DEF SEG = 1024
...
DEF SEG
```

يوضح هذا المثال استخدام العبارة بدون عنوان، وهذا يعيد القطاع إلى قطاع بيسك السريع للبيانات.

عملية تقليدية

توضح العملية عبارة DEF SEG. ابدأ بتحميل بيسك السريع.

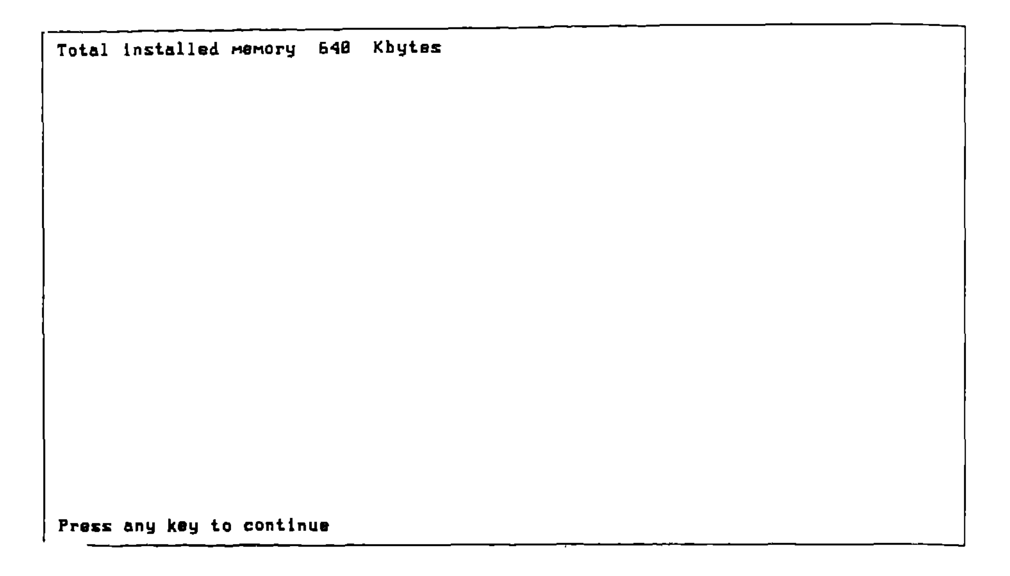
١ - اكتب البرنامج التالى :



```
'This program demonstrates the use of the DEF SEG statement.
'The program finds the amount of installed memory on the mother
'board, not extended or expanded, etc.
'The program should work on IBM PCs and full compatibles.

'find status in ram
DEF SEG = &H40
'memory size
totmem = 256 * PEEK(&H14) + PEEK(&H13)
CLS
PRINT "Total installed memory "; totmem; " Kbytes"
DEF SEG
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة DEF SEG فى البرنامج.



٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس المائة للاستمرار فى تسلسل التعلم.

الدرس الثانى والثلاثون

عبارات DEFDBL و DEFINT و DEFLNG و DEFSNG و DEFSTR

الوصف

تعرف هذه العبارات مجموعة من الحروف بأنها أنواع بيانات بسيطة. وتكوينها هو كمايلى :

```
DEFBDL letter range
DEFINT letter range
DEFLNG letter range
DEFSNG letter range
DEFSTR letter range
```

جزء letter range هو مدى الحروف الأبجدية التى تعرف بأنها من نوع معين. وتعرف DEFDBL مدى الحرف بأنه مزدوج الدقة. أما عبارة DEFINT فتعرف مدى الحرف بأنه صحيح. وتعرف عبارة DEFLNG أن مدى الحرف صحيح وطويل. أما عبارة DEFSNG فتعرف أن مدى الحرف فردى الدقة. وتعرف عبارة DEFSTR أن مدى الحرف من نوع السلسلة.

ما يعنيه هذا التعريف هو أن أى متغير يبدأ بحرف يقع فى المدى يكون من هذا النوع وذلك بدون عمل توضيح صريح لهذا المتغير. وتأخذ رموز توضيح النوع % و & و ! و # و $ أسبقية على عبارات DEFtype (وهى DEFDBL و DEFINT و DEFLNG و DEFSNG و DEFSTR). ولا تؤثر هذه العبارات على عناصر السجل. ولا تؤثر حالة الحرف فى شىء، أى إن ما يلى كله متساوى فى تعريفه :

```
DEFINT A-F
DEFINT a-F
DEFINT A-f
DEFINT a-f
```

ويمكن تعريف أكثر من مدى حرف واحد بأنه من نوع معين وذلك باستخدام الفاصلة كفاصل. مثال ذلك ما يلى :

```
DEFSTR a-e, q-s, x-z
```

وبمجرد تعريف مدى حرف بأنه من نوع معين فلا يمكن تغيير ذلك باستخدام أى عبارة أخرى من عبارات DEFtype.

التطبيقات

عبارات DEFtype تقدم وسيلة مريحة لتوضيح النوع بكميات كبيرة. وهى تشبه عبارات توضيح النوع فى لغة الفورتران. كما تؤثر عبارات DEFtype كذلك على دوال DEF FN وتوضيحات FUNCTION. وفيما يلى مثال لذلك :

```
DEFSNG P-T: DEFINT X-Z: DEFSTR A-F
..
X = 23: FirstMessage = "Press any key to continue "
```

لاحظ أن المتغير FirstMessage فى المثال لا يتطلب الحرف $ لتوضيح أنه من النوع الحرفى (السلسلة). فهو له نوع تقليدى من تعريف عبارة DEFSTR.

عملية تقليدية

توضح العملية عبارات DEFtype. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'The program demonstrates the use of the DEF type statements. The
'program defines several variables and assigns values to them.

DEFINT I-L: DEFSNG M-P: DEFDBL Q-T
DEFLNG A-D: DEFSTR X-Z

I = 128: L = 256
M = 22 / 7
Q = 22 / 7
A = 256000
Xtra = "This is a superfluous message in a superfluous sentence"
CLS
PRINT "DEFINT vars. ": I, L
PRINT "DEFSNG vars. ": M
PRINT "DEFDBL vars. ": Q
PRINT "DEFLNG vars. ": A
PRINT "DEFSTR vars. ": Xtra

                              Immediate

Main: <Untitled>        Context: Program not running          00017:028
```

٢ - نفذ البرنامج. لاحظ استخدام عبارات DEFtype فى البرنامج.

```
DEFINT vars.  128           256
DEFSNG vars.  3.142857
DEFDBL vars.  3.142857074737549
DEFLNG vars.  256000
DEFSTR vars. This is a superfluous message in a superfluous sentence






Press any key to continue
```

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الثالث عشر للاستمرار فى تسلسل التعلم.

الدرس الثالث والثلاثون

عبارة البعد DIM

الوصف

توضح عبارة البعد متغيراً على أنه منظومة وتخصص الذاكرة الخاصة له. وتكوينها هو كمايلى :

```
DIM variable (subscripts) AS type
```

جزء variable هو اسم متغير فى بيسك السريع. وجزء (subscript) اختيارى ويستخدم فى تعريف أبعاد المنظومة. وجزء AS type يعرف نوع المتغير. ويمكن أن يكون النوع نوعاً بسيطاً (INTEGER أو LONG أو SINGLE أو DOUBLE أو STRING) أو يكون من النوع الذى يعرفه المستفيد.

وجزء (subscript) له التكوين التالى :

```
(lowerlimit TO upperlimit)
```

ويحدد الحد الأدنى lowerlimit والحد الأعلى upperlimit حجم المنظومة. ويمكن أن تكون المنظومة متعددة الأبعاد. ويسمح بعدد أبعاد لايزيد عن 60 بعداً. وتوضح المنظومة التى لها أكثر من بعد واحد على النحو التالى :

```
DIM T (10.10)                     or
DIM T (1 TO 10, 1 TO 10)          or
DIM T (10, 1 TO 10)
```

والأمثلة السابقة متكافئة طالما أنه لم تنفذ أى عبارة OPTION BASE. ويتراوح مدى الدلائل من 32,768- إلى 32,767 مع السماح باستخدام أدلة سالبة فى المنظومة. وعندما يشار إلى احدى المنظومات بدليل يقع خارج مدى توضيحها تظهر رسالة خطأ تحدد أن الدليل يقع خارج المدى. وعندما تستخدم أى منظومة دون أن توضح فى عبارة بعد فإن أقصى قيمة لدليل هذه المنظومة تكون 10 وعندما توضح المنظومة فى عبارة بعد أكثر من مرة واحدة فتظهر رسالة خطأ توضح أن المنظومة قد سبق تحديد أبعاد لها بالفعل.

ويمكن لأى منظومة أن تكون استاتيكية $STATIC أو ديناميكية $DYNAMIC طبقاً لكيفية توضيحها. والمنظومة الاستاتيكية يتحدد لها ذاكرة عندما يترجم البرنامج. أما المنظومة الديناميكية فلا يكون لها ذاكرة محددة إلا عند التنفيذ. وفيما يلى سرد للطرق التى يمكن أن تكون المنظومة بها استاتيكية أو ديناميكية :

- عندما يوضح ثابت فى عبارة CONST أو يستخدم ثابت عددى فى عمل أبعاد المنظومة فتكون هذه المنظومة استاتيكية.
- المنظومات التى تحدد لها الأبعاد ضمنياً تكون منظومات استاتيكية.
- المنظومات التى تحدد لها الأبعاد باستخدام متغيرات كأدلة لها تكون منظومات ديناميكية.

ويبلغ أقصى حجم للمنظومة 64 كيلوبايت (65,535 بايت). وعندما تتعدى المنظومة هذا الحد تظهر رسالة خطأ تحدد أن المنظومة أكبر من اللازم. وتوضيح احدى المنظومات بأنها أكبر من 64 كيلوبايت يحدث عن طريق جعلها ديناميكية واستخدام خيار ah/ أثناء الترجمة.

ويمكن أن تشمل عبارة البعد بصورة اختيارية الخاصية SHARED. مثل ذلك ما يلى :

```
DIM SHARED variable AS type.
```

وهذا يجعل المتغير متاح الاتصال به عن طريق كل المقاطع. ويمكن أن توضح عبارة البعد أكثر من متغير واحد وذلك باستخدام فواصل تفصلها عن بعضها البعض.

```
DIM Var1 AS STRING, Var2 AS INTEGER, Var3 AS LONG
```

وتضع عبارة البعد قيماً ابتدائية لكل المتغيرات الموضحة. وتكون القيم الابتدائية اصفاراً للمتغيرات العددية وفراغات لمتغيرات السلاسل.

وتستخدم عبارة البعد فى توضيح متغيرات سجل. ويؤدى ذلك بتعريف نوع السجل بعبارات TYPE و END TYPE ثم توضيح متغير بعد ذلك من هذا النوع. وفيما يلى مثال لذلك :

```
TYPE RecType
Name AS STRING * 30
Address AS STRING * 80
END TYPE
DIM AddressRec AS RecType
```

التطبيقات

عبارة البعد هى احدى العبارات الأكثر قوة فى توضيح المتغيرات والمنظومات. وتعدد الجوانب لعبارة البعد فى بيسك السريع يجعلها قابلة للتطبيق فى تطبيقات عديدة. فيسمح توضيح المتغيرات التى يعدها المستفيد وتوضيح المتغيرات من النوع البسيط لأن يكون البرنامج أكثر ترتيباً وأسهل فى قراءته. وفيما يلى مثال لذلك :

```
DIM A$ AS STRING
DIM A AS INTEGER, B AS LONG, C AS DOUBLE
CONST MaxCol = 80, MaxRow = 25
DIM ScreenArray (MaxRow, MaxCol)
TYPE  NewLst
     LastPtr AS STRING * 8
     NextPtr AS STRING * 8
     ThisValue AS INTEGER
END TYPE
DIM CurrNode AS NewLst
```

عملية تقليدية

توضح هذه العملية استخدام عبارة البعد. ابدأ بتحميل بيسك السريع.

١ – أكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                            <Untitled>
'This program demonstrates the use of the DIM statement. The program
'declares a record variable and an array of that record variable.

TYPE BinType
   ItemName AS STRING * 25
   ItemNum AS INTEGER
   Qty AS INTEGER
END TYPE

DIM BinItem AS BinType
DIM BinList(10) AS BinType
S = 1
CLS
DO WHILE (UCASE$(Choice$) <> "Y") AND (S <= 10)
   INPUT "Enter Item name      : "; BinItem.ItemName
   INPUT "Item number          : "; BinItem.ItemNum
```

```
   INPUT "Quantity              : ": BinItem.Qty
   INPUT "Done ? (Y/N): ": Choice$
   BinList(S) = BinItem
   S = S + 1
LOOP

FOR Cnt = 1 TO S
   PRINT BinList(Cnt).ItemName, BinList(Cnt).ItemNum, BinList(Cnt).Qty
NEXT

                              Immediate
Main: <Untitled>      Context: Program not running              00016:001
```

٢- نفذ البرنامج وادخل البيانات التالية. لاحظ استخدام عبارة البعد.

```
Enter Item name     : ? Dust bins
Item number         : ? 100
Quantity            : ? 24
Done ? (Y/N): ? n
Enter Item name     : ? Dust Pans
Item number         : ? 200
Quantity            : ? 20
Done ? (Y/N): ? n
Enter Item name     : ? Bingo Cards
Item number         : ? 80
Quantity            : ? 2000
Done ? (Y/N): ? y
Dust bins                     100            24
Dust Pans                     200            20
Bingo Cards                   80             2000
                              0              0

Press any key to continue
```

٣ - ارجع إلى البرنامج. اختر برنامجاً جديداً دون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس المائة والخامس عشر للاستمرار في تسلسل التعلم.

الدرس الرابع والثلاثون

عبارة DO LOOP

الوصف

تمثل هذه العبارة أحد تكوينات التحكم فى المسار فى بيسك السريع. وتتسبب العبارة فى تكرار تنفيذ مجموعة عبارات فى البرنامج حتى تتحقق شروط معينة أو طالما أن نتيجة شروط معينة تكون صحيحة. ويمكن أن يأخذ تكوين هذه العبارة إحدى الصيغتين التاليتين :

التكوين الأول

```
DO
     statements
LOOP WHILE | UNTIL Boolean expression
```

التكوين الثانى

```
DO WHILE | UNTIL Boolean expression
     statements
LOOP
```

جزء statement فى كل من التكوينين عبارة عن عبارات من برنامج بيسك السريع. وتسمى مجموعة مثل هذه العبارات بكتلة (أو مجموعة) العبارات. وتوجد مجموعة العبارات داخل مكونات التحكم مثل WHILE WEND و FOR NEXT و FUNCTION و SELECT CASE و SUB و DEF FN متعددة الأسطر وغيرها. وبالطبع فإن البرنامج يكون مجموعة عبارات كذلك.

التطبيقات

تمثل هذه العبارة وسيلة تحكم متعددة الجوانب فى المسار. والتكوينان اللذان سبق وصفهما لهما تأثيرات مختلفة على تنفيذ البرنامج. ففى التكوين الأول تنفذ مجموعة العبارات أول مرة بغض النظر عما إذا تحقق الشرط أم لا. أى إن مجموعة العبارات تنفذ مرة واحدة على الأقل. وأما فى التكوين الثانى إذا لم تتحقق شروط تنفيذ الدورة عند البداية فلا تنفذ مجموعة العبارات على الاطلاق. أى إنه فى بعض الحالات يمكن أن تهمل مجموعة العبارات. وفيما يلى أمثلة لذلك :

مثال١

```
DO
  statement 1
  Q = Q + 1
LOOP WHILE (Q < 0)
```

مثال٢

```
DO WHILE (Q < 10)
  statement 1
  Q = Q + 1
LOOP
```

فى المثال الأول تنفذ الدورة مرة واحدة على الأقل بغض النظر عن قيمة Q. وفى المثال التالى له يمكن أن تهمل الدورة كلية إذا ما تعدت قيمة Q المقدار 10. وكل من الصيغتين لها استخداماتها ويعتمد اختيار أى منهما على التطبيق وعلى تكوين البرنامج. استخدام UNTIL بدلاً من WHILE هو موضوع اختيار فقط طبقاً لتكوين تعبير بوليان.

عملية تقليدية

تسترجع فى هذه العملية البرنامج الذى سبق حفظه فى الدرس السادس والثمانين وعليك أن تلاحظ عبارة DO LOOP. ابدأ بتحميل بيسك السريع.

١ - اختر فتح البرنامج من قائمة File واضغط على Tab للانتقال إلى الدليل واختر الملف LRTRIM. BAS من الدليل مستخدماً مفاتيح الأسهم ثم اضغط على مفتاح الادخال.

٢ - اضغط على Shift-F2 لتنقيح عبارة FUNCTION. عدل عبارة DO LOOP كما هو مبين فى القائمة التالية. احذف كذلك السطر الذى يسبق عبارة DO مباشرة.

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                LRTRIM.BAS
'This program demonstrates the use of LTRIM$ and RTRIM$ functions
'This program is modified from Module 75

DECLARE FUNCTION GCh$ (VCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
DIM VCh AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"
VCh = "EDCQ"
P = "Enter selection"
Selection$ = GCh$("EDCQ", 24, 1, P)
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
```

```
FUNCTION GCh$ (UCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
  DIM Choice AS STRING
  P = LTRIM$(RTRIM$(P))
  Choice = " "
  DO WHILE (INSTR(UCh, UCASE$(Choice)) = 0)
     LOCATE x, y
     PRINT P;
     INPUT ; Choice
     LOCATE x, (y + LEN(P))
     PRINT Choice;
  LOOP
  GCh$ = Choice
END FUNCTION
```

٣ – نفذ البرنامج ولاحظ استخدام عبارة DO LOOP فى البرنامج.

```
E)dit / C)reate / D)elete / Q)uit ?
Enter selection?
```

٤ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر Save As من قائمة File واكتب DO-LOOP. BAS ثم أكتب اسم الملف واحفظ هذا البرنامج على أنه برنامج نصى.

٥ – من قائمة File اختر برنامجاً جديداً واكتب N لإخلاء الشاشة.

٦ – انتقل إلى الدرس الثامن للاستمرار فى تسلسل التعلم.

الدرس الخامس وثلاثون

عبارة DRAW

الوصف

ترسم هذه العبارة شيئاً على الشاشة طبقاً لما هو مذكور بواسطة تعبير سلسلة المؤشرات. وتكوينها هو كما يلي :

```
DRAW string exp
```

جزء string exp هو سلسلة لغة ماكرو يصف كيفية الاستمرار في عمل الرسم. ومكونات لغة الماكرو هي كما يلي :

- أوامر حركة نقطة البداية.
- بادئات لحركة نقطة البداية.
- أوامر معامل مقياس وزاوية دوران وزاوية تحويل ولون للرسم وأوامر للألوان.
- أوامر تنفيذ سلاسل جزئية.

أوامر حركة نقطة البداية هي كما يلي :

الجـزء	الوصــــف
U(n)	ينقل نقطة البـداية عـدد n مـن الوحــدات. والوحــدة هـي نقطة رسـم إلا إذا ما تحدد غير ذلك عند وضع أمر معامل المقياس.
D(n)	ينقل نقطة البداية عدد n من الوحدات لأسفل.
L(n)	ينقل نقطة البداية عدد n من الوحدات لليسار.
R(n)	ينقل نقطة البداية عدد n من الوحدات لليمين.
E(n)	ينقل من الركن السفلى الأيسر إلى الركن العلوى الأيمن عدد n من الوحدات.
F(n)	ينقل من الركن العلوى الأيسر إلى الركن السفلى الأيمن عدد n من الوحدات.
G(n)	ينقل من الركن العلوى الأيمن إلى الركن السفلى الأيسر عدد n من الوحدات.
H(n)	ينقل من الركن السفلى الأيمن إلى الركن العلوى الأيسر عدد n من الوحدات.
M x, y	ينتقل إلى نقطة مطلقة أو نسبية احداثياتها x, y على الشاشة. عندما يسبق x اشارة فتكون الحركة نسبية للموضع الحالى وإلا فإنها تكون مطلقة. ويرسم خط من الموقع الحالى إلى x, y.

بادئات حركة نقطة البداية

فيما يلى الأوامر التى يمكن استخدامها كبادئات لأى أمر من أوامر الحركة.

البادئة	الوصف
B	ينقل نقطة البداية ولايرسم أى نقاط
N	ينقل ويرسم إلا أنه يعود إلى الموقع الأصلى.

أوامر TURN و ROTATE و SCALE و COLOR :

تتعامل الأوامر التالية مع زاوية الدوران وزاوية التحويل ومعامل التكبير أو التصغير (المقياس) واختيار اللون فى عبارة DRAW.

الجزء	الوصف
TA n	يتسبب فى دوران الرسم بزاوية مقدارها n درجة حيث تقع n بين 360- و 360. يستخدم أمر TA مع دالة VARPTR$ مثل : DRAW"TA=" + VARPTR$ (Angle)
A n	يحدد زاوية دوران الرسم. تقع n بين 0 و 3 حيث 0 يعنى درجة 0 و 1 يعنى 90 درجة و 2 يعنى 180 درجة و 3 يعنى 270 درجة.
S n	يحدد معامل المقياس حيث تقع n بين 1 و 255. ويحدد ذلك حجم وحدات الحركة.
C n	تحدد أن اللون هو n. ارجع إلى الدروس 29 و 98 و 124 لمناقشة الألوان والأعداد.
P n1, n2	تحدد اللون لتلوين الشكل من داخله، n1 هو اللون المستخدم فى التلوين و n2 هو لون الحدود.
V	تجعل كل أوامر DRAW ترسم باستخدام المنطق بدلاً من الاحداثيات الواقعية.

تنفيذ السلاسل الجزئية

تسمح عبارة DRAW بتنفيذ سلاسل أخرى من داخل السلسلة التى تنفذ حالياً. والتكوين هو كما يلى :

```
"X" + VARPTR$(string)
```

من الممكن كذلك تنفيذ سلاسل جزئية أخرى.

التطبيقات

تمثل هذه العبارة وسيلة قوية تنفذ لغة ماكرو لرسم أشياء على الشاشة. ويمكن أن تستخدم عبارة DRAW إذا ما كانت لديك امكانيات رسومات وعرض ملونة فقط. وفيما يلى أمثلة لعبارة DRAW :

مثال١

```
SCREEN 1
DRAW "E20 F20 E20 F20"
```

مثال٢

```
Box$ =  "U40 R40 D40 L40 E20 P1,1"
DRAW Box$
```

مثال٣

```
DRAW "M 100,100" + Box$
```

مثال٤

```
DRAW "M  -100,100" + Box$
```

عملية تقليدية

هذه العملية عبارة عن توضيح بسيط لعبارة DRAW. استمر فيها إذا كانت لديك امكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                           F1=Help
                                  DRAW.BAS
'This program demonstrates the use of the DRAW statement.

SCREEN 1
DRAW "c1 u5 r5 d5 l5 "
DRAW "bm+20,0 u5 r5 d5 l5 "
DRAW "Bm-8,0 d10"
DRAW "bm-5,5 r10"
DRAW "bm+15,0 u30 l35"
xp = POINT(0): yp = POINT(1)
DRAW "d30 f10 r15 e10"
DRAW "bm=" + VARPTR$(xp) + ",=" + VARPTR$(yp)


FOR Cnt = 1 TO 12
   DRAW "c2 ne5 "
   DRAW "bm+3,0"
NEXT

                                  Immediate

Main: DRAW.BAS        Context: Program not running                 00008:020
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة DRAW فى البرنامج.

٣ – ارجع إلى البرنامج واحفظه كملف نصى تحت اسم ملف DRAW.BAS.

٤ – انتقل إلى الدرس السادس عشر للاستمرار فى تسلسل التعلم.

الدرس السادس والثلاثون

عبارة END

الوصف

تستخدم عبارة END فى انهاء برنامج البيسك أو فى انهاء اجراء أو فى انهاء مجموعة من مجموعات البيسك. وتكوين انهاء كل منها هو كما يلى :

END DEF | FUNCTION | SELECT | IF | SUB | TYPE

الجـزء	الوصـــف
END	كلمة من كلمات البيسك المحجوزة. واستخدامها فى حد ذاتها يحدد نهاية برنامج البيسك وتقوم باغلاق كل الملفات وتعود إلى بيئة التشغيل.
DEF	كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انهاء عبارة DEF FN.
FUNC-TION	كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انهاء اجراء دالة FUNCTION.
SELE-CT	كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انهاء مكون SELECT CASE.
IF	كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انهاء مكون IF.. THEN.. LSE على هيئة مجموعة.
SUB	كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انهاء اجراء SUB.
TYPE	كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انهاء عبارة تعريف أنواع يحددها المستفيد.

التطبيقات

يمكن استخدام عبارة END فى انهاء البرنامج من أى موقع فى البرنامج. فإذا حذفت كلمة END فيفترض مترجم البيسك السريع أن البرنامج ينتهى عندما لاتكون هناك أسطر أخرى فى البرنامج لتنفيذها. وفيما يلى بعض الأمثلة :

مثال١

```
'Program to print 10 random numbers
RANDOMIZE TIMER      ' seed the random number generator with the timer value
FOR i = 1 TO 10
  PRINT INT((20-0+1) * RND+0) " ":
NEXT i
END   ' of program
```

يبين ذلك استخدام END فى انهاء البرنامج. يمكن أن تحذف عبارة END دون أى تاثير على تنفيذ البرنامج.

مثال٢

```
DEF FNTokenize%(InpStr$)
  FNTokenize% = 0
  FOR I = 1 to LEN(InpStr$)
    Token% = Token% + ASC(MID$(InpStr$,I,1))
  NEXT I
  FNTokenize% = Token%
END DEF
```

يبين هذا المثال تعريف دالة العودة إلى token التى تنتج بواسطة اضافة قيم ASCII لكل الرموز الموجودة فى المؤشر. تستخدم عبارة END فى انهاء التعريف.

مثال٣

```
'demonstrates the SELECT CASE statement
PRINT "M)aster / T)ransaction / R)eport / mA)in menu ?"
INPUT "Enter choice ", C$
SELECT CASE C$
   CASE "M"
      GOSUB MasterFileProcess
   CASE "T"
      GOSUB TransFileProcess
   CASE "R"
      GOSUB ReportProcess
   CASE "A"
      GOTO RetMenu2
   CASE ELSE
      PRINT "Invalid selection ..."
END SELECT
```

يبين هذا المثال كيفية استخدام عبارة SELECT CASE فى بناء تكوين متعدد الفروع واستخدام END CASE في انهاء التكوين.

مثال٤

```
FUNCTION  LowerCase$ (InpStr$)
  FOR  I = 1 TO LEN(InpStr$)
    IF INSTR ("ABCDEFGHIJKLMNOPQRSTUVWXYZ", MID$(InpStr$,I,1)) THEN
      MID$(InpStr$,I) = CHR$(ASC(MID$(InpStr$,I,1)+32))
    END IF
  NEXT I
  LowerCase$ = InpStr$
END FUNCTION
```

يستخدم هذا المثال عبارتى END IF و END FUNCTION. وتحول الدالة سلسلة إلى حروف صغيرة.

مثال٥

```
SUB . DisplayPrompt(PromptStr$)
   LOCATE 24,1: PRINT PromptStr$;
END SUB
```

هذا مثال لاستخدام END SUB.

مثال٦

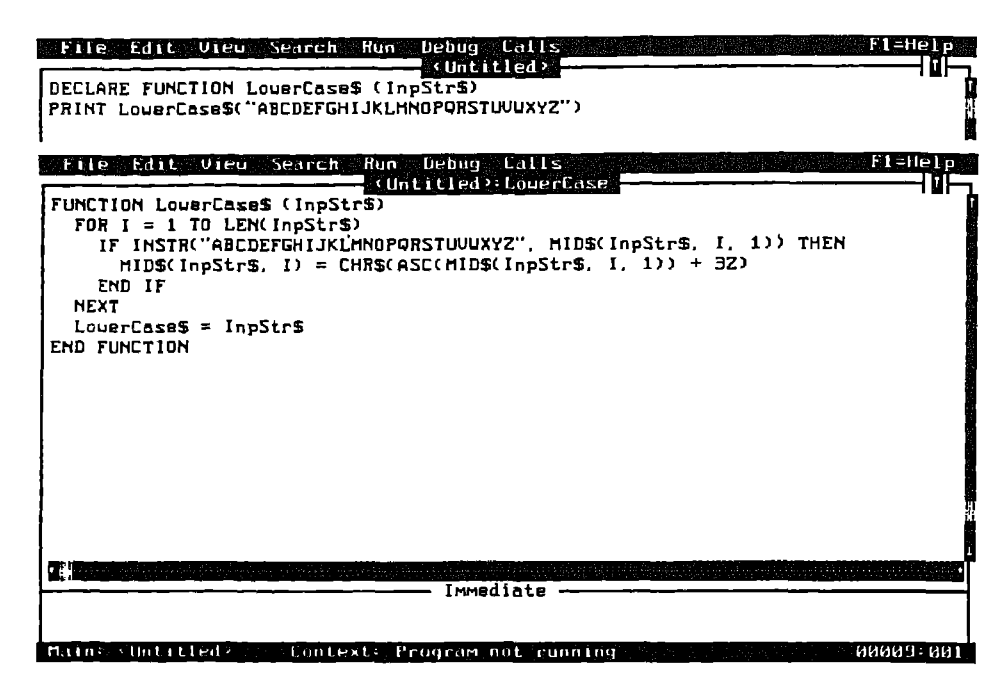
```
TYPE  WindowParms
   WindowData AS STRING[4000]
  LastX       AS INTEGER
  LastY       AS INTEGER
END TYPE
```

يبين هذا المثال استخدام END TYPE.

عملية تقليدية

يوضح مثال البرنامج التالى استخدام عبارة END. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
<Untitled>
DECLARE FUNCTION LowerCase$ (InpStr$)
PRINT LowerCase$("ABCDEFGHIJKLMNOPQRSTUVWXYZ")

File  Edit  View  Search  Run  Debug  Calls                    F1=Help
<Untitled>:LowerCase
FUNCTION LowerCase$ (InpStr$)
  FOR I = 1 TO LEN(InpStr$)
    IF INSTR("ABCDEFGHIJKLMNOPQRSTUVWXYZ", MID$(InpStr$, I, 1)) THEN
      MID$(InpStr$, I) = CHR$(ASC(MID$(InpStr$, I, 1)) + 32)
    END IF
  NEXT
  LowerCase$ = InpStr$
END FUNCTION

Immediate
Main: <Untitled>     Context: Program not running          00009:001
```

٢ – اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ استخدام عبارة هذا الفصل فى البرنامج.

```
abcdefghijklmnopqrstuvwxyz



Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج.

٤ – اختر Save من قائمة File واكتب END.BAS كاسم للملف. حدد أن الملف من النوع النصى واحفظ البرنامج.

٥ – اخل الشاشة عن طريق اختيار برنامج جديد من قائمة File.

٦ – انتقل إلى الدرس المائة والسابع والأربعون للاستمرار فى تسلسل التعلم.

الدرس السابع والثلاثون

عبارة ENVIRON ودالة $ENVIRON

الوصف

عبارة ENVIRON ودالة $ENVIRON تتعاملان مع متغيرات بيئة DOS وذلك مثل PATH و $PROMPT، وتكوينهما هو كما يلى :

```
ENVIRON string expression
ENVIRON$(environment string)
ENVIRON$(n)
```

عبارة ENVIRON: جزء expression هو السلسلة التى يراد عملها كجزء من بيئة DOS. ويشمل تعبير السلسلة متغير البيئة والقيمة الجديدة التى تحدد له. ويوجد متغير البيئة بالفعل فإذا لم يكن موجودا فإنه يضاف إلى القائمة. وتعبير السلسلة له الصيغة التالية :

```
parameter=text
```

جزء parameter هو متغير بيئة كما أن جزء text هو القيمة الجديدة المحددة لهذا المتغير.

دالة $ENVIRON : تعيد كلاً من صيغتى التكوين لدالة $ENVIRON السلسلة الحالية المصاحبة لمتغير البيئة المطلوب. عندما تستخدم سلسلة البيئة فتكون السلسلة هى المؤشر المطلوب له معلومات. عندما تستخدم n فتقرأ قائمة متغيرات البيئة ويعود المؤشر رقم n والنص المصاحب له. مثال ذلك تعطى ("PATH") $ENVIRON أعداد المسار الحالى وتعطى (1) $ENVIRON أعداد أول متغير بيئة.

التطبيقات

تستخدم عبارة ENVIRON ودالة $ENVIRON فى الاتصال بمتغيرات بيئة DOS وتعديلها. والاستخدام محدد بمتطلبات التطبيق. وفيما يلى أمثلة لذلك :

مثال١

```
ENVIRON "PATH=C:\;C:\DOS"
```

مثال٢

```
PRINT ENVIRON$("PATH")
```

مثال٣

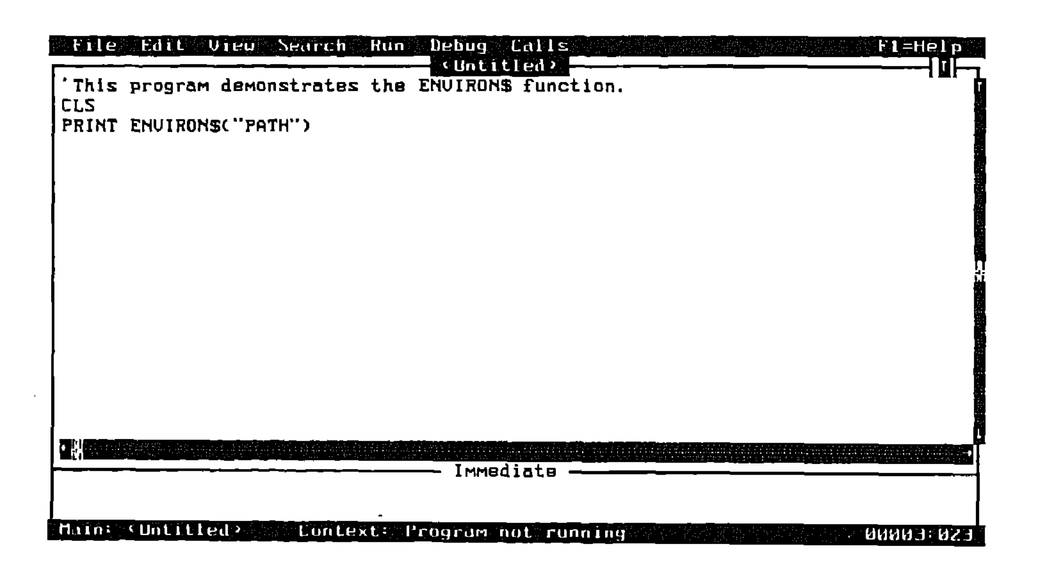
```
PRINT ENVIRON$(10)
```

عندما يوجد المؤشر فى قائمة متغير البيئة فتعيد ENVIRON$ سلسلة فارغة. ويظل تأثير عبارة ENVIRON أثناء تنفيذ البرنامج فقط وبمجرد انتهاء البرنامج تعود متغيرات بيئة DOS مرة أخرى إلى المكان الذى كانت فيه من قبل.

عملية تقليدية

تمثل هذه العملية توضيحاً بسيطاً لدالة ENVIRON$. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
'This program demonstrates the ENVIRON$ function.
CLS
PRINT ENVIRON$("PATH")
```

٢ - لاحظ استخدام دالة ENVIRON$ فى البرنامج. ارجع إلى البرنامج واخل الشاشة دون أن تحفظ البرنامج.

٣ - انتقل إلى الدرس السابع عشر للاستمرار فى تسلسل التعلم.

الدرس الثامن والثلاثون

دالـة EOF

الوصف

تستخدم دالة EOF فى التأكد من شرط انتهاء الملف، وتعيد قيمة عددية صحيحة. وتكوينها هو كما يلى :

```
EOF(filenum)
```

جزء filenum هو الرقم المحدد للملف فى عبارة OPEN، وتحدد القيمة 1- بأنها صحيحة (وهى نهاية الملف المصاحب لرقم الملف)، كما تحدد أى قيمة أخرى بأنها خاطئة لاختبار نهاية الملف. فإذا ما كانت هناك محاولة لادخال بيانات بعد الوصول إلى نهاية الملف فتظهر رسالة خطأ بأنه لا ادخال لبيانات.

وعند الاستخدام مع وحدة اتصالات فتعيد دالة EOF نتيجة شرط انتهاء الملف على النحو التالى:

ASCII FILE : تصبح نهاية الملف صحيحة فى هذه الحالة عندما يتم استقبال Ctrl-Z وطالما أنه صحيح وحتى يغلق الملف.

BINARY FILE : تصبح نهاية الملف صحيحة فى هذه الحالة عندما يكون صف المدخلات (Loc(Filenum) = 0).

ولا يمكن استخدام EOF على الوحدات التالية : SCRN و KYBRD و CONS و LPTx، حيث x هو رقم صحيح.

التطبيقات

إستخدم دالة EOF كلما كان مطلوباً عمل اتصال بالملف، ويمنع ذلك من القراءة بعد الملف وحدوث خطأ وقت التشغيل فى البرنامج وفيما يلى بعض الامثلة.

مثال١

```
OPEN "Sales.Dat" FOR INPUT AS #3
 . .
WHILE NOT EOF(3)
    . .
    . .
WEND
```

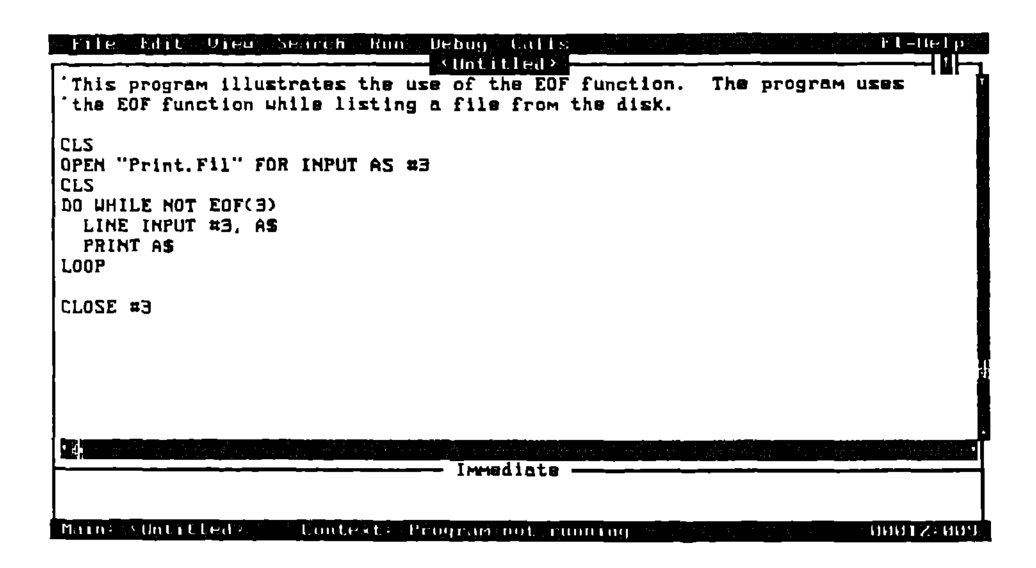
مثال٢

```
OPEN "Sess.Log" FOR RANDOM AS #1
..
IF EOF(1) THEN
..
ELSE
..
```

عملية تقليدية

توضح هذه العملية استخدام دالة EOF. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج. لاحظ استخدام دالة EOF فى البرنامج.

```
Earth Moving Equipment        12            $120,000
Farm Equipment                22            $85,000
Farm 22   $85.0
SOS





Press any key to continue
```

٣ - اختر برنامجاً جديداً من قائمة File واختر عدم حفظ البرنامج.

٤ - انتقل إلى الدرس الثامن والسبعين للاستمرار فى تسلسل التعلم.

الدرس التاسع والثلاثون

عبارة ERASE

الوصف

تلغى عبارة ERASE مواقع الذاكرة المحددة لمنظومة ديناميكية أو تعيد وضع القيم الابتدائية لعناصر منظومة استاتيكية وتكوينها هو كما يلى :

```
ERASE array
```

جزء array هو المنظومة التى تتأثر. فإذا كانت المنظومة ديناميكية فتخلى الذاكرة المستخدمة بواسطة المنظومة، أما إذا كانت المنظومة استاتيكية فتوضع قيم ابتدائية لعناصرها. عندما تكون المنظومة استاتيكية فتوضع قيم صفر للعناصر العددية وقيم فراغات لعناصر السلاسل. يمكن وضع قيم ابتدائية لأكثر من منظومة واحدة كما يمكن الغاء مواقع أكثر من منظومة واحدة وذلك عن طريق استخدام فواصل تفصل بين المنظومات فى عبارة ERASE مثل ما يلى :

```
ERASE Array1, Array2, Array3.
```

التطبيقات

عبارة ERASE هى وسيلة أخرى لإدارة ذاكرة وقت التنفيذ فى بيسك السريع. ويمكن استخدام عبارة REDIM مع المنظومات الديناميكية بدون استخدام عبارة ERASE وذلك لأن عبارة REDIM ترعى اعادة عمل الأبعاد للمنظومة الديناميكية. وفيما يلى بعض الأمثلة :

```
'$STATIC
DIM Art(320,680)
...
ERASE Art
```

يوضح المثال السابق عبارة ERASE المستخدمة فى اعادة وضع قيم ابتدائية فى المنظومة الاستاتيكية Art.

```
'$DYNAMIC
DIM PictureArr(225,420)
..
REDIM PictureArr(320,680)
..
ERASE PictureArr
```

يوضح المثال السابق استخدام REDIM بدون تدخل من ERASE واستخدام ERASE فى الغاء الذاكرة المحددة لـ PicturArr.

عملية تقليدية

توضح هذه العملية استخدام عبارة ERASE. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                            <Untitled>
'This program demonstrates the use of the ERASE statement. The program
'uses a dynamic array to find the minimum and maximum values of the two
'lists. The ERASE statement is used to get rid of the array
'before redimensioning.

Max = 15
DIM A(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A(Cnt)
  NEXT
  RETURN

FindMinMax:
 MinVal = A(1): MaxVal = A(1)
 FOR Cnt = 2 TO Max
   IF MinVal > A(Cnt) THEN
     MinVal = A(Cnt)
   END IF
   IF MaxVal < A(Cnt) THEN
     MaxVal = A(Cnt)
   END IF
 NEXT
 RETURN

Start:
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "First pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  READ Max
  ERASE A
  DIM A(Max)
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "Second pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  ERASE A
```

```
DATA 12,23,33,43,1,56,98,656,323,44,9,80,67,54,10
DATA 10
DATA 8,89,76,54,23,32,12,4,33,54
```

```
Immediate
Main: <Untitled>     Context: Program not running     00054:011
```

٢ - نفذ البرنامج، لاحظ استخدام ERASE فى البرنامج، أعد وضع أبعاد المنظومة بدون استخدام عبارة REDIM.

```
First pass
Minimum of array:  1        Maximum of array:  656
Second pass
Minimum of array:  4        Maximum of array:  89







Press any key to continue
```

٣ - ارجع إلى البرنامج واخل الشاشة دون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الثانى والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الأربعـون

دالـتا ERDEV و ERDEV$

الوصف

تستخدم دالـتا ERDEV و ERDEV$ فى الحصـول على حالة خطأ محددة فى الوحدة. وتكوينهما هو كما يلى :

```
ERDEV
ERDEV$
```

ولا تحـتـاج أى من الدالتين لوجـود مـؤشـر ولا يقـبل أى مـؤشـر. وتعطى دالة ERDEV رمز الخطأ للوحدة التى انتجت الخطأ. كما تعطى دالة ERDEV$ اسم الوحـدة التى حدث عندها الخطأ. والقيمة التى تعود من دالة ERDEV تكون على هيئة شفرة بت مع ضغط أدنى ثمانية بت برمز الخطأ فى DOS. ويوجد المزيد من فك قيم شفرة البت فى الدليل التقنى لنظام DOS.

التطبيقات

تستخدم دالتا ERDEV و ERDEV$ فى الحصـول على مـعلومـات عن شروط الخطأ فى الوحدات مـثل مشـغل الأقراص أو بوابات الاتصالات. وفيـما يلى أمـثلة لاسـتخدام ERDEV و ERDEV$.

مثال١

```
ON ERROR GOTO PrintErrAndStop
..
PrintErrAndStop:
PRINT "Error code on ";ERDEV$;" is ":ERDEV STOP
```

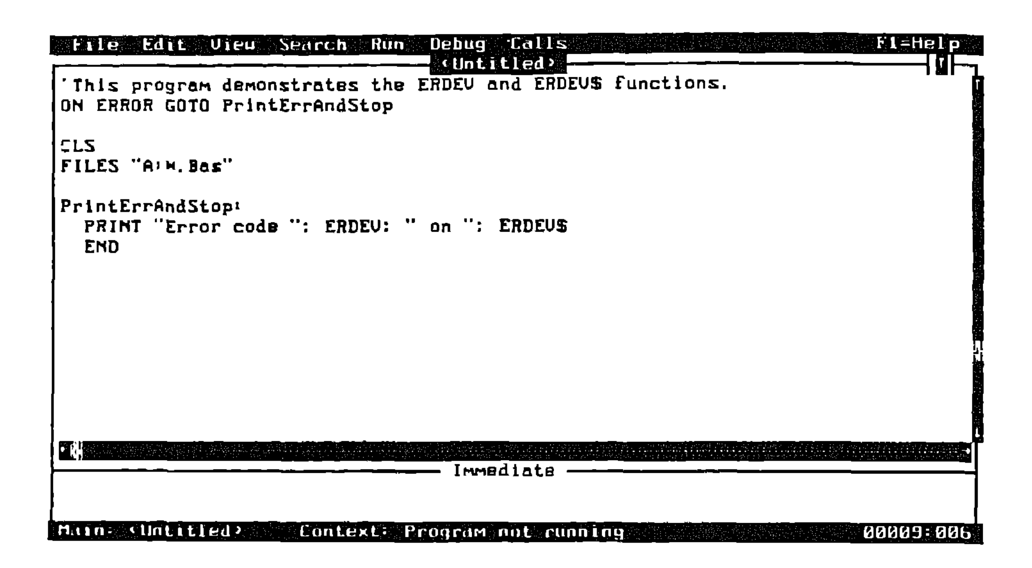
مثال٢

```
ON ERROR GOSUB SaveError
..
SaveError:
DevError = ERDEV: DevName = ERDEV$
..
```

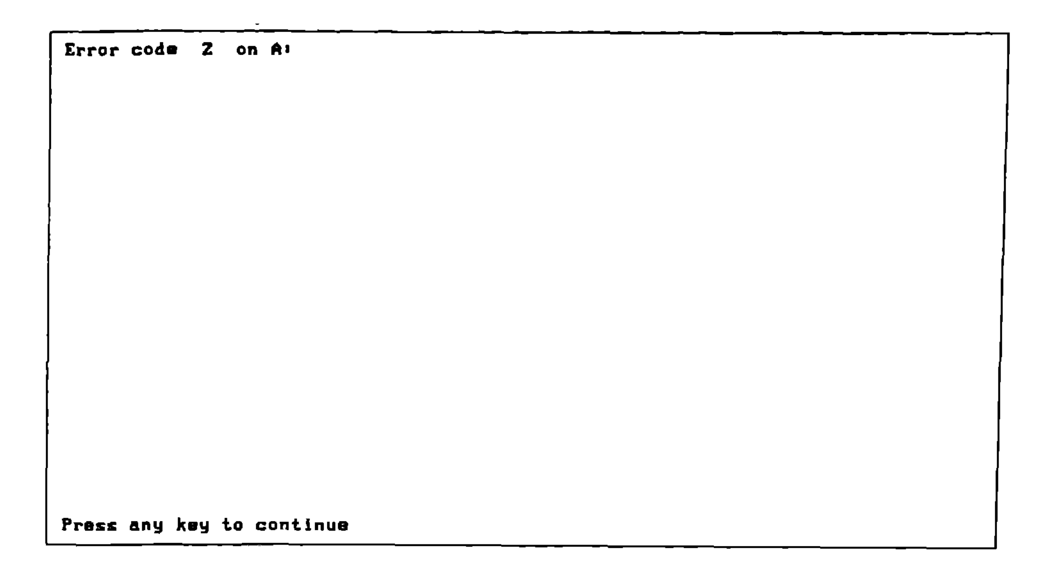
عملية تقليدية :

توضح هذه العملية استخدام دالتى ERDEV و ERDEV$. يحاول البرنامج أن يعرض الدليل فى المشغل A ويظهر الخطأ لأن باب مشغل الأقراص يكون مفتوحاً. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



٢ – نفذ البرنامج مع ترك باب مشغل الأقراص A مفتوحاً. نتيجة البرنامج هى كما يلى. لاحظ استخدام دالتى ERDEV و ERDEV$ فى البرنامج.



٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس التاسع والخمسين للاستمرار فى تسلسل التعلم.

الدرس الحادى والأربعون

دالتا ERR و ERL وعبارة ON ERROR GOTO

الوصف

دالة ERR : تعيد دالة ERR رمز الخطأ لاحدث خطأ وقع فى البرنامج، وعندما تختبر للحصول على النتائج الفورية بعد الخطأ فتعيد دالة ERR رمز الخطأ المصاحب لهذا الخطأ، وتكوينها هو كما يلى :

```
ERR
```

ولا يوجد مؤشر لدالة ERR، فهى تعيد قيمة صحيحة تقع بين 1 و 255، لا يمكن استخدام دالة ERR فى الطرف الأيسر من عبارة التحديد.

دالة ERL : تعيد دالة ERL رقم السطر الذى حدث عنده أحدث خطأ، وتكوينها هو كمايلى:

```
ERL
```

ولايوجد مؤشر لدالة ERL، فإذا لم توجد أى أرقام أسطر فى البرنامج فتعيد دالة ERL الرقم 0، ولاتعيد اسم السطر، ودالة ERL مثل دالة ERR لاتستخدم فى الطرف الأيسر لعبارة التحديد.

عبارة ON ERROR GOTO : تمكن عبارة ON ERROR GOTO من معالجة الخطأ داخل البرنامج، وتكوينها هو كما يلى :

```
ON ERROR GOTO line
```

جزء GOTO line من العبارة يحدد رقم أول سطر أو اسم السطر المصاحب لمقطع معالجة الخطأ، وعندما يحدث أحد الأخطاء يقفز التحكم فى البرنامج إلى السطر المحدد ويستمر التنفيذ من عند هذا السطر، ورقم السطر 0 يلغى المقدرة على اصطياد الخطأ، وعندما تنفذ عبارة ON ERROR GOTO 0 داخل معالج الخطأ فيطبع البرنامج رسالة الخطأ وينتهى تشغيل البرنامج كلية.

ومعالج الخطأ ليس برنامجاً فرعياً، ولاتستخدم تكوينات SUB و DEF FN و FUNC-TION كمعالجات للخطأ.

التطبيقات :

تقدم دوال ERR و ERL وعبارة ON ERROR GOTO آلية قوية تسمح باصطياد خطأ وقت التشغيل فى البرنامج. وتنفيذ هذه الوسائل بسيط ويمكن على ذلك أن يستخدمه المبتدئون. ويقلل هذا النوع من اصطياد الأخطاء من الحاجة إلى العديد من عبارات GOTO وعبارات IF..THEN..ELSE وعبارات GOSUB المستخدمة فى التحكم فى تداخل البرنامج مع المستفيد ومع وحدات المدخلات والمخرجات. وتتولى بيئة وقت تشغيل برنامج بيسك السريع اصطياد الأخطاء المهمة وتسمح بتبسيط شيق للبرنامج. وفيما يلى بعض الأمثلة :

دالة ERR :

مثال١

```
IF ERR = 53 THEN PRINT "File not found. Please try again."
```

مثال٢

```
WHILE ERR = 0
   statement
   statement 2
WEND
```

مثال٣

```
NewError = ERR
PRINT NewError
```

دالة ERL :

مثال١

```
IHandleErrors:
  NewError = ERR
  IF NewError = 12  THEN PRINT NewError;" at ";ERL
END
```

مثال٢

```
NewErrorLine = ERL
WRITE #2 "Error occurred at .";NewErrorLine
```

عبارة ON ERROR GOTO

مثال١

```
'Program to print the names and addresses
'All global variables, no subroutines
DEFINT A-G
ON ERROR GOTO 1000
 ...
 ...
1000 PRINT ERR,"Error occurred. What is going on?"
END
```

مثال٢

```
ON ERROR GOTO CrashHelmet
  ..
  ..
CrashHelmet:
    ...
    ...
RESUME
```

عملية تقليدية :

توضح هذه العملية استخدام دالة ERR وعبارة ON ERROR GOTO ودالة ERL. ابدأ بتحميل بيسك السريع.

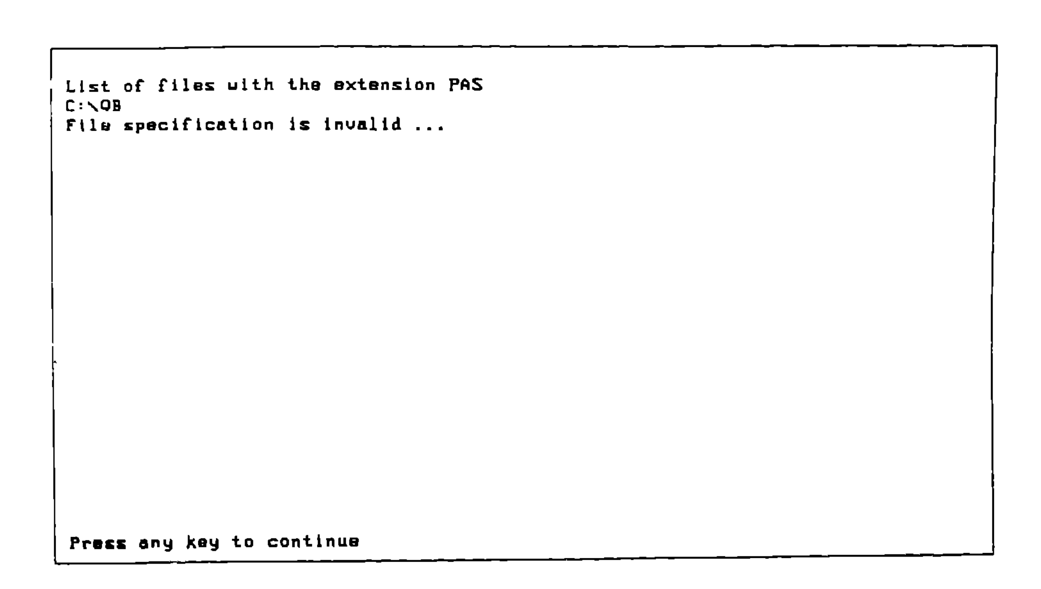
١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This is a demonstration of the ERR and ON ERROR GOTO statements.
'The program attempts to display files from the logged directory.
'The error code 53 means "file not found"; translated in this context,
'it means the file specification is invalid.


ON ERROR GOTO ErrorHandler
CLS
PRINT : PRINT "List of files with the extension PAS "
FILES "*.PAS"
END
ErrorHandler:
  IF ERR = 53 THEN
     PRINT "File specification is invalid ..."
  ELSE
     PRINT "Don't know what happened .."
  END IF
END
                              Immediate

Main: <Untitled>      Context: Program not running              00001:001
```

٢ - نفذ البرنامج. لاحظ استخدام ERR و ERL و ON ERROR GOTO في البرنامج. اضغط على أى مفتاح للعودة للبرنامج.



٣ - غير فى البرنامج عن طريق تغيير PAS إلى BAS. ارجع إلى البرنامج ملاحظاً أن الأخطاء قد تم تصحيحها.

```
List of files with the extension BAS
C:\QB
SAMPLE  .BAS      REMLINE .BAS      SORTDEMO.BAS      TORUS   .BAS
DEMO1   .BAS      DEMO2   .BAS      DEMO3   .BAS      INCH2CM .BAS
BOX     .BAS      BOX2    .BAS      PRINT   .BAS      STRING  .BAS
IFTHEN  .BAS      CASE    .BAS      ASC     .BAS      ULCASE  .BAS
LRTRIM  .BAS      ABS     .BAS      RANDOM  .BAS      DOLOOP  .BAS
PLAY    .BAS      ONEVENT .BAS
 858880 Bytes free




Press any key to continue
```

٤ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٥ – انتقل إلى الدرس الثانى والأربعين للاستمرار فى تسلسل التعلم.

الدرس الثانى والأربعون

عبارة ERROR

الوصف

تسمح عبارة ERROR لك بتعريف رموز خاصة بك للأخطاء أو بماحاكاة خطأ بيسك عندما يستدعى برمز خطأ موجود بالفعل. وتكوينها هو كما يلى :

```
ERROR integer expression
```

جزء integer expression تقع قيمته بين 1 و 255. تحاكى عبارة ERROR شرط الخطأ المحدد عندما يكون *التعبير العددى الصحيح* عبارة عن رمز خطأ موجود. وإلاً فان عبارة ER-ROR تعرف *التعبير العددى الصحيح* كرمز جديد للخطأ. وتعيد دالة ERROR رموز الخطأ سابقة التعريف والتى يعرفها المستفيد. واحدى الطرق الجيدة لتعريف رموز خطأ خاصة بك هى البدء من 255 والعمل لأسفل وهذا يسمح بالتوافقية مع صيغ بيسك السريع المستقبلية التى يمكن أن تكون بها رموز جديدة للخطأ معرفة داخلها.

التطبيقات :

عبارة ERROR هى حليف قوى مع الآليات الأخرى لمعالجة الأخطاء والمتوفرة فى بيسك السريع. المقدرة على تعريف رموز خطأ جديدة محددة للتطبيق هى احدى طرق التعود على بيئة وقت التنفيذ. وفيما يلى بعض الأمثلة :

مثال١

```
ERROR 12            'simulates the error code 12
```

مثال٢

```
ERROR 53            'simulates the error code 53
```

مثال٣

```
'assigns new error code 255 that can be defined in the
'error handling routine
```

عملية تقليدية :

توضح هذه العملية استخدام عبارة ERROR. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                        F1=Help
                              <Untitled>
'This is a demonstration of the ERROR statement.  The program uses the
'ERROR statement to define new error conditions and to trap occurrences
'of that error.

ON ERROR GOTO Eh
CLS : PRINT
PRINT : PRINT "This program demonstrates the ERROR statement "
INPUT "Do you wish to continue (Y/N)  "; Yn$
  IF Yn$ = "Y" OR Yn$ = "y" THEN
     ERROR 255
  ELSE
     IF Yn$ = "N" OR Yn$ = "n" THEN
     ERROR 254
     ELSE ERROR 253
     END IF
  END IF
END

Eh:
 IF ERR = 255 THEN
    PRINT "What for? There is nothing going on here ..."
 ELSE
    IF ERR = 254 THEN
       PRINT "Why not? You don't know what you are missing!"
    ELSE PRINT "What? Don't you want to play along ..."
    END IF
 END IF

END

------------------------------ Immediate ------------------------------

Main: <Untitled>        Context: Program not running            00036:003
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة ERROR فى البرنامج. اكتب Y واضغط على مفتاح الادخال. اضغط على أى مفتاح للعودة إلى البرنامج.

```
This program demonstrates the ERROR statement
Do you wish to continue (Y/N)  ? y
What for? There is nothing going on here ...





Press any key to continue
```

٣ - اختر New من قائمة File وأكتب N لاخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والتسعين للاستمرار في تسلسل التعلم.

الدرس الثالث والأربعون

عبارة EXIT

الوصف :

تستخدم عبارة EXIT فى ترك دورة أو دالة أو برنامج فرعى تركا نهائياً. وتكوينها هو كمايلى:

```
EXIT structure
```

جزء structure هو أى تكوين فى بيسك السريع. وفيما يلى قائمة بصيغ مختلفة لعبارة EXIT:

```
EXIT FOR
EXIT LOOP
EXIT DEF
EXIT FUNCTION
EXIT SUB
```

كل من صيغ عبارة EXIT يناسب غرضاً محدداً فقط مذكوراً على النحو التالى. واستخدام هذه الصيغ دون تمييز لغرض استخدام كل منها يتسبب فى خطأ وقت الترجمة.

عبارة EXIT FOR : تتسبب هذه العبارة فى الخروج من دورة FOR، ويستمر تنفيذ البرنامج من السطر الذى يلى دورة FOR مباشرة. وعندما تكون دورات FOR متداخلة فيستمر تنفيذ البرنامج من عند السطر الذى يلى دورة FOR الحالية مباشرة.

عبارة EXIT LOOP : تتسبب هذه العبارة فى الخروج من دورة DO LOOP. ويستمر تنفيذ البرنامج من السطر الذى يلى دورة DO LOOP مباشرة. وعندما تكون دورات DO LOOP متداخلة فيستمر تنفيذ البرنامج من عند السطر الذى يلى دورة DO LOOP الحالية مباشرة.

عبارة EXIT DEF : تتسبب هذه العبارة فى الخروج من دالة DEF. ويستمر تنفيذ البرنامج من عند السطر الذى استدعى DEF FN.

عبارة EXIT FUNCTION : تتسبب هذه العبارة فى الخروج من دالة FUNCTION. ويستمر تنفيذ البرنامج من عند السطر الذى استدعى FUNCTION.

عبارة EXIT SUB : تتسبب هذه العبارة فى الخروج من برنامج فرعى SUB، ويستمر تنفيذ البرنامج من عند السطر الذى استدعى SUB.

التطبيقات :

توفر دالة EXIT مرونة فى ترك التكوين تركاً نهائياً عندما تؤدى الشروط إلى ذلك. مثال ذلك تنتهى دورة FOR عندما يصل عداد الدورة إلى الحد المحدد فى عبارة FOR فإذا لم يكن هذا بسبب عبارة EXIT FOR فيجبر البرنامج على الاستمرار فى تنفيذ الدورة حتى عندما يتحقق شرط معين ومثل هذا الاستمرار غير مناسب، وفيما يلى بعض الأمثلة :

مثال١

```
FOR Tc = 1 TO 25
     ...
     IF (Mens > 0) THEN EXIT FOR
NEXT
```

مثال٢

```
DO WHILE NoPrint
     ..
     IF OutPaper THEN EXIT LOOP
LOOP
```

مثال٣

```
DEF FNReName(NewName AS STRING)
     ..
     IF FileNotFound THEN EXIT DEF
END DEF
```

عملية تقليدية :

توضح هذه العملية استخدام EXIT، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                   <Untitled>
'This program demonstrates the use of the EXIT statement. The program
'generates the period of a pendulum given the length, and exits
'the function when the length is less than or equal to 0.

CONST pi = 3.14159
GOTO Start

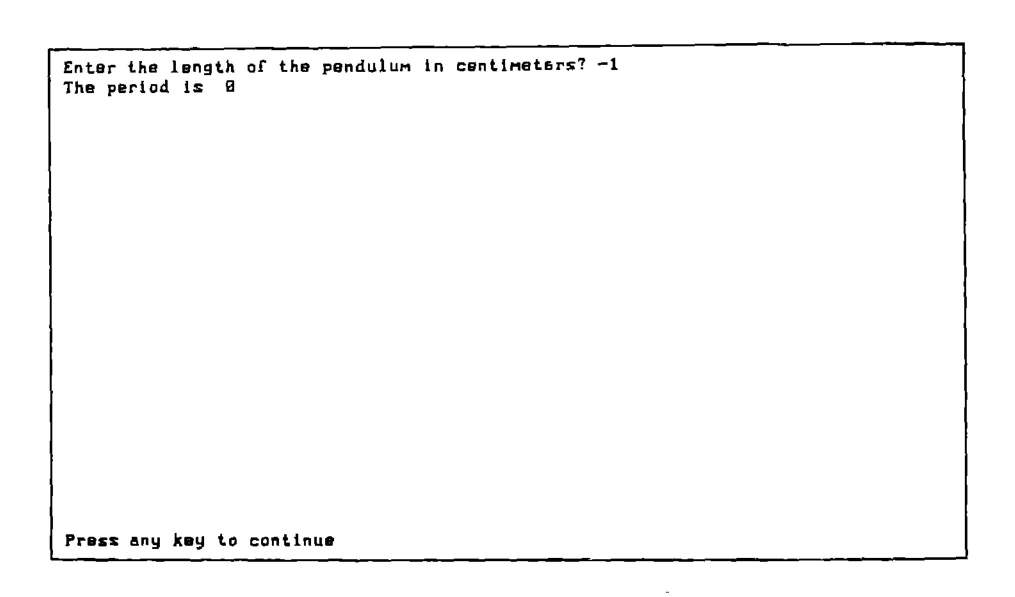
DEF FnPeriod! (Length)
   IF Length <= 0 THEN EXIT DEF
   FnPeriod! = (2 * pi) * SQR(Length / 981)
END DEF

Start:
  CLS
  INPUT "Enter the length of the pendulum in centimeters": Length
  PRINT "The period is ": FnPeriod!(Length)

                                   Immediate

Main: <Untitled>      Context: Program not running                    00016:044
```

٢ – نفذ البرنامج. اكتب 1- واضغط على مفتاح الإدخال. لاحظ استخدام EXIT فى البرنامج.



٣ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٤ – انتقل إلى الدرس الحادى والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الرابع والأربعون

دالــة EXP

الوصف

تحسب دالة EXP الدالة الأسية لتعبير عددى معين، وتكوينها هو كما يلى :

```
EXP(numeric expression)
```

والقيمة التى تعيدها الدالة هى e (أساس اللوغاريتم الطبيعى) مرفوعاً لقوة التعبير العددى، تكون قيمة التعبير العددى أقل من 88.02969 وعندما يتعدى التعبير العددى هذه القيمة فتظهر رسالة خطأ السريان الزائد "overflow". وتحسب القيمة التى تعود كقيمة لها دقة فردية بصورة تقليدية، وعندما يكون التعبير العددى له دقة مزدوجة فتحسب دالة EXP بدقة مزدوجة كذلك.

التطبيقات

تعيد دالة EXP القوة الأسية المحددة، قوة للأساس e، والاستخدام محدد بالتطبيق المبرمج فقط.

عملية تقليدية

توضح العملية التالية استخدام دالة EXP. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

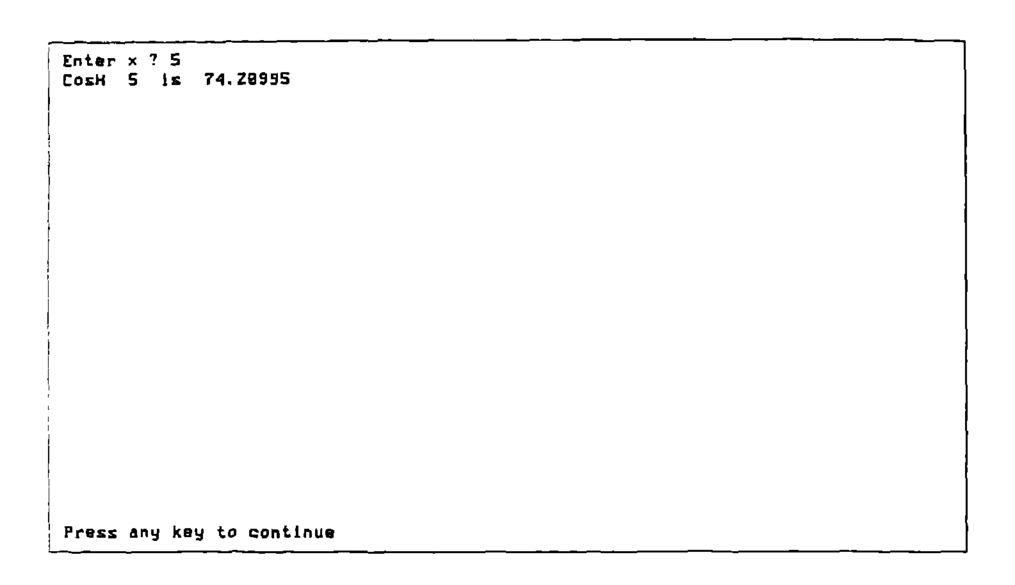
```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              EXP.BAS
'This program computes the hyperbolic trigonometric COSH function.
'It is calculated CosH x = e^x + e^-x / 2
CLS
INPUT "Enter x ": x
Temp! = EXP(x) + EXP(-x)
PRINT "CosH ": x: " is ": Temp / 2



─────────────────────────────── Immediate ──────────────────────────────

 Main: EXP.BAS        Context: Program not running             00002:024
```

٢ – نفذ البرنامج. اكتب 5 كاستجابة للملقن. لاحظ استخدام دالة EXP فى البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.



```
Enter x ? 5
CosH  5  is  74.20995

Press any key to continue
```

٣ – من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الثانى والثمانين للاستمرار فى تسلسل التعلم.

الدرس الخامس والأربعون

عبارة FIELD

الوصف

تحدد عبارة FIELD مواقع للمتغيرات فى الذاكرة الاحتياطية لملف الاتصال المباشر. وتكوينها هو كما يلى :

```
FIELD # filenum. field width AS string var.
```

جزء filenum هو رقم يحدد للملف فى عبارة OPEN، وجزء field width هو حجم الحقل فى السجل. وجزء string var هو الموقع الذى تخزن فيه البيانات قبل كتابتها فى الملف أو بعد قراءتها من الملف ز ويمكن أن تعطى تعريفات لحقول متعددة مفصولة عن بعضها البعض بواسطة فواصل. ويجب ألا يزيد اجمالى أطوال الحقول عن حجم السجل المحدد فى عبارة OPEN. فاذا ما حدث ذلك فتظهر رسالة تفيد بسريان زائد للحقل "FIELD overflow". يمكن تنفيذ عبارات FIELD متعددة فى نفس الوقت وتظل نشطة فى نفس الوقت كذلك.

لاتستخدم متغير سلسلة معرفاً كحقل فى عبارة INPUT إذا ما أردت أن تستخدمه كحقل نظراً لأنه يعطى نتائج غير متوقعة.

التطبيقات

تسمح عبارة FIELD بتحكم اضافى على البيانات المخزنة فى ملف وذلك بتعريف مجموعة رموز للمعالجة. وهى أكثر فائدة عند استخدام ملفات غير نصية أى ملفات بها بيانات مختلطة. وفيما يلى بعض الأمثلة :

```
CONST BinID = 10, BinQty = 10, ReorderPt = 5, RecSize = 25
OPEN "Inven.Dat" FOR RANDOM AS #1 LEN = RecSize
FIELD #1 BinID AS BinNum$, BinQty AS QtyOnHand, ReorderPt AS RP
```

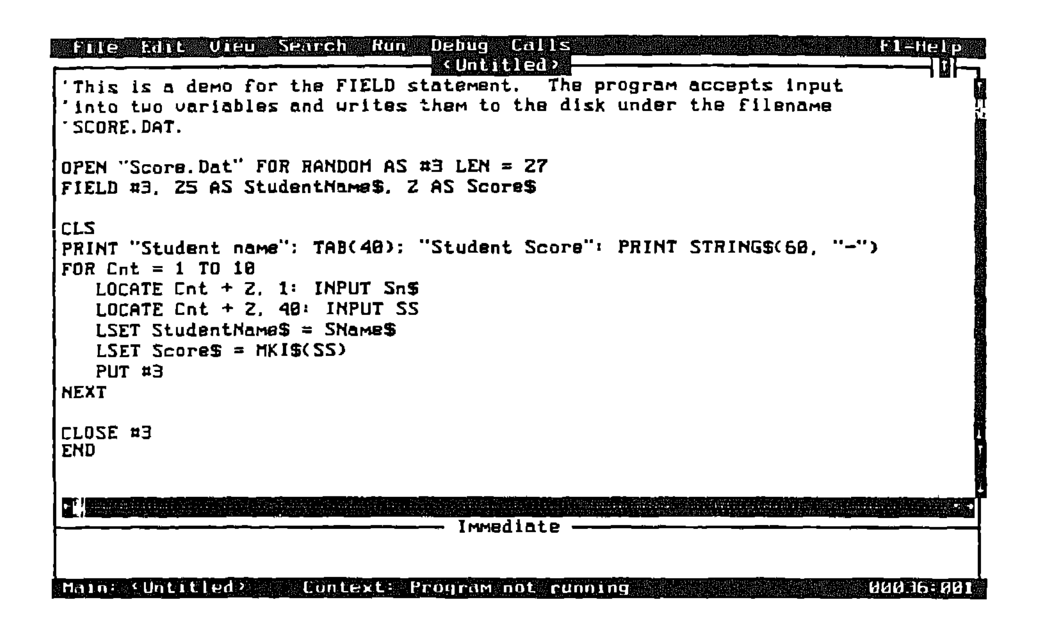
تعرف عبارة CONST فى المثال السابق أحجام الحقول وتعد عبارة OPEN الملف وتعرف عبارة FIELD محتويات الحقل.

```
CONST FirstN = 30, MidInit = 1, LastName = 30, RecSize = 61
OPEN "Name.Lst" FOR OUTPUT AS #3 LEN = RecSize
FIELD #3 FirstN AS FirstName$, MidInit AS MidInitial$, LastName AS LName$
```

عملية تقليدية :

توضح هذه العملية استخدام عبارة FIELD. ابدأ بتحميل بيسك السريع.

١ – اختر New من قائمة File واكتب البرنامج التالى :



٢ – نفذ البرنامج. لاحظ استخدام عبارة FIELD فى البرنامج فى اعداد الحقول الفردية فى الملف. اكتب البيانات التالية واضغط على مفتاح الادخال بعد كل ادخال. بعد 10 ادخالات فى كل حقل ينتهى البرنامج.

```
Student name                              Student Score
----------------------------------------------------------------
? Mary Jane                               ? 34
? Judy O.                                 ? 43
? Susie P.                                ? 55
? QBert                                   ? 54
? Mark S. Man                             ? 44
? Henry                                   ? 89
? Pipsqueak                               ? 89
? Nid                                     ? 67
? Sid                                     ? 78
? Pel                                     ? 78

Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الخامس والثمانين للاستمرار فى تسلسل التعلم.

الدرس السادس والأربعون

دالــة FILEATTR

الوصف

تعطى دالة FILEATTR معلومات عن ملف مفتوح. وتكوينها هو كما يلى :

```
FILEATTR(filenum, attribute)
```

جزء filenum هو رقم الملف المستخدم فى عبارة OPEN عندما تم فتح هذا الملف. وجزء at- tribute أما أن يكون 1 أو 2. وعندما يكون 1 فتعيد دالة FILEATTR رقماً محدداً الحالة التى فتح فيها الملف على النحو التالى :

نتيجة الدالــة	حالة الملــف
1	مدخلات
2	مخرجات
4	عشوائى
8	اضافة
32	ثنائى

وعندما تكون قيمة attribute هى 2 فتعيد دالة FILEATTR رقم معالجة ملف DOS. ورقم معالجة ملف DOS هو رقم داخلى يستخدمه نظام DOS فى تتبع الملف المفتوح.

التطبيقات

تستخدم دالة FILEATTR فى الحصول على معلومات عن ملف بيسك السريع الذى سبق فتحه. والبرنامج الفرعى الذى ليس له اتصال بعبارة OPEN والذى فتح الملف فعلاً يمكنه أن يحصل على معلومات عن الملف ويتخذ قرارات عن كيفية تشغيله. والاستخدامات الأخرى تعتمد على التطبيق نفسه. وفيما يلى أمثلة لاستخدام FILEATTR :

مثال١

```
OPEN Fil$ FOR RANDOM AS #2
..
IF FILEATTR(2,1) = 32 THEN GOSUB ProcessBinary
```

مثال٢

```
SUB CustomerFileProcess(FNum AS INTEGER)
     FMode = FILEATTR(FNum, 1)
     ..
END SUB
```

عملية تطبيقية

تفتح هذه العملية سلسلة من الملفات وتطبع المعلومات التى تأتى من دالة FILEATTR. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
'Illustration of FILEATTR function.
CLS
PRINT "Program opens several files and prints information about them"
OPEN "Print.fil" FOR INPUT AS #1
OPEN "002" FOR OUTPUT AS #2
OPEN "003" FOR RANDOM AS #3
OPEN "004" FOR APPEND AS #4
OPEN "005" FOR BINARY AS #5
PRINT : PRINT
PRINT "Filename   ": "File number   ": "File mode   ": "File handle"
PRINT "Print.Fil  ": "1             ": FILEATTR(1, 1): SPC(10): FILEATTR(1, 2)
PRINT "002        ": "2             ": FILEATTR(2, 1): SPC(10): FILEATTR(2, 2)
PRINT "003        ": "3             ": FILEATTR(3, 1): SPC(10): FILEATTR(3, 2)
PRINT "004        ": "4             ": FILEATTR(4, 1): SPC(10): FILEATTR(4, 2)
PRINT "005        ": "5             ": FILEATTR(5, 1): SPC(10): FILEATTR(5, 2)
```

٢ – نفذ البرنامج. لاحظ استخدام FILEATTR فى البرنامج. تكون مخرجات البرنامج كمايلى:

```
Program opens several files and prints information about them


Filename   File number  File mode  File handle
Print.Fil  1            1          5
002        2            2          6
003        3            4          7
004        4            8          8
005        5            32         9










Press any key to continue
```

٣ - ارجع إلى البرنامج. اختر New ولا تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الثامن عشر للاستمرار فى تسلسل التعلم.

الدرس السابع والأربعون

عبارات FILES و CHDIR و MKDIR و RMDIR

الوصف

عبارة FILES : تستخدم عبارة FILES فى طباعة أسماء الملفات الموجودة على قرص معين أو فى دليل معين. وتكوينها هو كما يلى :

```
FILES file specification
```

جزء file specification اختيارى ويذكر لعبارة FILES المكان الذى تبحث فيه عن الملفات ونوع الملفات التى تبحث عنها. وعندما يحذف هذا الجزء فتسرد عبارة FILES كل أسماء الملفات الموجودة على المشغل الحالى أو فى الدليل الحالى. رموز wild-card النمطية فى DOS يمكن استخدامها فى مواصفات الملف. ويجب أن توضع مواصفات الملف بين علامتى تنصيص مزدوجتين إذا ما كانت حرفية.

وتخدم عبارة FILES عند استخدامها فى أحد البرامج كوسيلة لجعل البرنامج صديقاً للمستفيد. وفيما يلى بعض الأمثلة لعبارة FILES :

```
FILES
```

تطبع هذه العبارة كل الملفات الموجودة على مشغل الأقراص الحالى أو فى الدليل الحالى.

```
FILES "*.DAT"
```

تطبع هذه العبارة كل الملفات التى لها اتساع DAT.

```
FILES "A:*.SCR"
```

تطبع هذه العبارة كل الملفات الموجودة على مشغل الأقراص : A ولها اتساع SCR..

```
FILES "C:\*.*"
```

تطبع هذه العبارة كل الملفات الموجودة فى دليل الجذر على القرص : C.

وتطبع عبارة FILES الدليل الحالى كعنوان بغض النظر عن مواصفات الملف.

عبارة CHDIR : تغير عبارة CHDIR الدليل التقليدى على مشغل محدد. وتكوينها هو كما يلى :

```
CHDIR path specification
```

جزء path specification هو جزء المسار الجديد للدليل التقليدى. ويمكن أن يكون طوله حتى 64 رمزاً كحد أقصى. وتكوين المسار هو مثل تكوين اسماء مسارات DOS. ويجب أن يوضع وصف الملف بين علامتى تنصيص مزدوجتين إذا كان حرفياً.

تستخدم عبارة CHDIR فى تغيير الدلائل أثناء تنفيذ البرنامج. وفيما يلى بعض الأمثلة :

```
CHDIR "\NEW.DAT"
CHDIR "\BASIC\QB.400\MAINT"
```

عبارة MKDIR : تنتج عبارة MKDIR دليلاً جديداً طبقاً لمواصفات المسار المعطاة وتكوينها هو كما يلى :

```
MKDIR path specification
```

يقدم جزء path specification اسم الدليل الجديد. ويمكن أن يكون طوله حتى 128 حرفاً كحد أقصى. وتكون مواصفات المسار متطابقة مع تكوين أسماء المسار فى DOS. ويجب أن توضع مواصفات الملف بين علامتى تنصيص مزدوجتين إذا كان حرفياً.

تستخدم عبارة MKDIR فى انتاج أدلة جديدة أثناء تنفيذ البرنامج. وفيما يلى بعض الأمثلة:

```
MKDIR "C:\BASIC"
MKDIR "C:\PASCAL"
MKDIR "\TEMP"
```

عبارة RMDIR : تحذف عبارة RMDIR دليلا طبقاً للمسار المحدد. وتكوينها هو كما يلى:

```
RMDIR path specification
```

جزء path specification هو للدليل المراد حذفه. ويمكن أن يكون طوله حتى 128 حرفاً كحد أقصى. ويحذف الدليل إذا لم يكن فيه أى ملفات. وتتبع مواصفات المسار نفس تكوين اسماء المسارات فى DOS. ويجب أن توضع مواصفات الملف بين علامتى تنصيص مزدوجتين إذا كان حرفياً.

وتستخدم عبارة RMDIR فى حذف أدلة موجودة أثناء تنفيذ البرنامج. وفيما يلى بعض الأمثلة :

```
RMDIR "C:\BASIC\TEMP"
RMDIR "\SALES\OLD.DAT"
RMDIR "\SCRATCH"
```

التطبيقات

عبارات FILES و CHDIR و MKDIR و RMDIR مفيدة جداً فى الاتصال بالقرص والتعامل معه اثناء تنفيذ البرنامج. والاستخدامات محدودة بالتطبيق المبرمج فقط.

عملية تقليدية

توضح العملية التالية استخدام عبارات FILES و CHDIR و MKDIR و RMDIR. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                              <Untitled>
'This program demonstrates the FILES, CHDIR, MKDIR, and RMDIR statements.

CLS
PRINT : PRINT
PRINT "First we create a directory so that we can change directories !"
PRINT "The directory will be called A:\TEMPQB.ILL."
PRINT "The directory will be created in drive A:,"; so: make: sure: you; ""
PRINT "have a formatted diskette in that drive with at least one file on it."

MKDIR "A:\TEMPQB.ILL"
PRINT
PRINT "Now we change to the root directory and list that directory."
INPUT "Press Enter to continue"; C$

CHDIR "A:\"
FILES "A:\"

INPUT "Press Enter to continue"; C$

INPUT "Press Enter to continue"; C$
PRINT : PRINT "Now we change to the new directory, and list that directory."
INPUT "Press Enter to continue"; C$
CHDIR "A:\TEMPQB.ILL"
FILES "A:\TEMPQB.ILL\*.*"

INPUT "Press Enter to continue"; C$
PRINT
PRINT "Now we change back to the root and delete the directory we created."
PRINT "And we will show the directory to prove it !"
INPUT "Press Enter to continue"; C$
CHDIR "A:\"
RMDIR "A:\TEMPQB.ILL"
FILES "A:\"

                              Immediate

Main: <Untitled>      Context: Program not running                 00034:011
```

٢ - نفذ البرنامج واتبع تعليماته. لاحظ تأثير عبارات FILES و CHDIR و MKDIR و RMDIR على حالة القرص.

```
First we create a directory so that we can change directories !
The directory will be called A:TEMPQB.ILL.
The directory will be created in drive A:, so make sure you
have a formatted diskette in that drive with at least 1 file on it.

Now we change to the root directory and list that directory.
Press Enter to continue?
A:\
README  .TXT      TEMPQB  .ILL<DIR>
 346112 Bytes free
Press Enter to continue?

Now we change to the new directory, and list that directory.
Press Enter to continue?
A:\TEMPQB.ILL
      .   <DIR>          ..  <DIR>
 346112 Bytes free
Press Enter to continue?

Now we change back to the root and delete the directory we created.
And we will show the directory to prove it !
Press Enter to continue?
```

```
The directory will be created in drive A:, so make sure you
have a formatted diskette in that drive with at least one file on it.

Now we change to the root directory and list that directory.
Press Enter to continue?
A:\
README  .TXT      TEMPQB  .ILL<DIR>
 346112 Bytes free
Press Enter to continue?

Now we change to the new directory, and list that directory.
Press Enter to continue?
A:\TEMPQB.ILL
      .   <DIR>          ..  <DIR>
 346112 Bytes free
Press Enter to continue?

Now we change back to the root and delete the directory we created.
And we will show the directory to prove it !
Press Enter to continue?
A:\
README  .TXT
 347136 Bytes free

Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس التسعين للاستمرار فى تسلسل التعلم.

الدرس الثامن والأربعون

دالـــة FIX

الوصف

تعيد دالة FIX الجزء الصحيح بعد الغاء الكسر العشرى من التعبير العددى. وتكوينها هو كما يلى :

```
FIX(numeric expression)
```

الوصـــف	الجـــزء
كلمة بيسك سريع محجوزة	FIX
العدد أو القيمة التى يلغى كسرها العشرى	numeric expression

التطبيقات

تستخدم دالة FIX عندما تكون هناك حاجة إلى حذف الكسر العشرى من عدد له دقة فردية أو دقة مزدوجة. مثال ذلك :

```
PRINT FIX(12.33)
```

المخرجات هى : 12

```
PRINT FIX(228.211)
```

المخرجات هى : 228

```
PRINT FIX(-811)
```

المخرجات هى : 811-

```
PRINT FIX(-9009.0001)
```

المخرجات هى : 9009-

عملية تقليدية

توضح العملية التالية دالة FIX. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
CLS
PRINT "FIX(-123.05) returns ": FIX(-123.05)
PRINT "INT(-123.05) returns ": INT(-123.05)
PRINT "FIX(123.05) returns ": FIX(123.05)
PRINT "INT(123.05) returns ": INT(123.05)

                         Immediate

Main: <Untitled>      Context: Program not running          00005:047
```

٢ – نفذ البرنامج، لاحظ مخرجات البرنامج واستخدام دالة FIX.

```
FIX(-123.05) returns -123
INT(-123.05) returns -124
FIX(123.05) returns  123
INT(123.05) returns  123

Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F واضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس السادس والستين للاستمرار فى تسلسل التعلم.

الدرس التاسع والأربعون

عبارة FOR.. NEXT

الوصف

العبارات التى تسمح لك بالتحكم فى مسار البرنامج تسمى تكوينات تحكم control structures. وعبارة FOR.. NEXT فى البيسك هى مثل تكوين التحكم هذا. تنفذ عبارات البرنامج التى تقع داخل حدود دورة FOR.. NEXT عدداً معيناً من المرات. وتكوينها هو كما يلى:

```
FOR counter = numeric expression  TO
              numeric expression [STEP numeric expression]

   program statements
NEXT counter
```

وأجزاء التكوين هى كما يلى :

الجـــزء	الوصـــف
FOR	كلمة بيسك محجوزة.
counter	متغير عددى. لايمكن أن يكون عنصراً من عناصر سجل أو منظومة فى سجل.
numeric expression	تعبير عددى صحيح. يمكن أن تكون القيمة عدداً مزدوج الدقة. وتفضل القيم العددية الصحيحة لتحقيق السرعة.
TO	كلمة بيسك محجوزة.
STEP	جزء اختيارى للتحكم فى خطوة زيادة العداد counter إذا كانت هناك حاجة إلى ذلك. وهى كلمة بيسك محجوزة.
NEXT	كلمة بيسك محجوزة تحدد نهاية العبارات التى تقع داخل دورة FOR.. NEXT.

التداخل هو طريقة كتابة لدورات FOR.. NEXT بحيث إنه توجد دورات FOR.. NEXT أخرى داخل تكوين FOR.. NEXT. وعندما يحدث تداخل لعبارات FOR.. NEXT فيجب أن يوجد متغير عداد فردى لكل عبارة من هذه العبارات المتداخلة. وفيما يلى مثال يوضح بعض الطرق المختلفة لتكوين دورة FOR.. NEXT.

```
FOR Cnt1 = 1 TO 10
   ...
     FOR Cnt2 = 1 TO 10 STEP 2
         program statements
         ...
          FOR Cnt3 = 10 TO 1 STEP -1
                ...
                FOR Cnt4 = 1 TO LEN(Str00$)
                ...
                NEXT Cnt4
          NEXT Cnt3
     NEXT Cnt2
NEXT Cnt1
(or)
NEXT Cnt4,Cnt3,Cnt2,Cnt1
```

لاحظ الأربعة أسطر لعبارات NEXT والتى يمكن استبدالها بعبارة NEXT واحدة. لاحظ كذلك ترتيب العدادات فى عبارة NEXT واحدة.

التطبيقات

عبارة FOR.. NEXT مفيدة جداً عندما يتطلب البرنامج تكرار تنفيذ مجموعة من التعليمات عدداً معيناً من المرات. وامكانية تداخل عبارات FOR.. NEXT فى بعضها تعتبر ميزة. وعادة ما تستخدم قيمة العداد counter داخل مجموعة FOR.. NEXT فى الحسابات. وفيما يلى مثال لذلك.

```
k = 2
FOR i = 1 TO 10
  PRINT i, k*i
NEXT i
```

```
 1            2
 2            4
 3            6
 4            8
 5            10
 6            12
 7            14
 8            16
 9            18
 10           20

Press any key to continue
```

عملية تقليدية

توضح العملية التالية استخدام دورة FOR.. NEXT فى برنامج. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
CLS
INPUT "Enter rate per hour ", Rate
INPUT "Enter number of hours allowed per week ", Hours%
INPUT "Enter number of weeks worked ", Weeks%
PRINT : PRINT STRING$(40, "-")
PRINT "Week": SPACE$(10): "Cumulative Wages": PRINT STRING$(40, "-")
FOR Cnt = 1 TO Weeks%
  PRINT Cnt: TAB(14): ((Rate * Hours%) * Cnt)
NEXT

                              Immediate

Main: <Untitled>    Context: Program not running             00010:001
```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج. اكتب القيم 10.635 و 40 و 12 ولاحظ المخرجات.

```
Enter rate per hour 10.635
Enter number of hours allowed per week 40
Enter number of weeks worked 12

----------------------------------------
Week          Cumulative Wages
----------------------------------------
 1            425.4
 2            850.8
 3            1276.2
 4            1701.6
 5            2127
 6            2552.4
 7            2977.8
 8            3403.2
 9            3828.6
 10           4254
 11           4679.4
 12           5104.8



Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - من قائمة File اختر Save. اكتب FORNEXT.BAS كاسم للملف ثم حدد صيغة الملف على أنها نصية واحفظ البرنامج. اضغط على Alt-F ثم اضغط على N لإخلاء الشاشة.

٥ - انتقل إلى الدرس الثامن والخمسين للاستمرار فى تسلسل التعلم.

الدرس الخمسون

دالــة FRE

الوصف

تعطى دالة FRE كمية الذاكرة المتاحة فى مناطق مختلفة أثناء وقت التنفيذ. وتكوينها هو كمايلى:

```
FRE(number)
FRE[string]
```

نعيد دالة FRE، مع عدد كمؤشر، الذاكرة المتاحة فى المناطق التالية وذلك طبقاً لقيمة العدد.

العدد	القيمــة
-1	حجم أكبر منظومة ليست سلسلة يمكن أن يعد لها أبعاد
-2	الكمية المتاحة لمكان الرصة.
other	حجم المجموعة التالية المتاحة لمخزن سلسلة.

والقيم التى تعود تكون كلها بالبايت.

وعندما يكون المؤشر سلسلة أو تعبير سلسلة فتعطى دالة FRE كمية التخزين المتاحة للسلسلة. كما أنها تضغط كذلك المخزن المتاح للسلسلة فى مجموعة واحدة.

التطبيقات

يمكن أن تعزز بحكمة دالة FRE تنفيذ البرنامج بطرق عديدة. وفيما يلى منطقتان لعمل ذلك: التأكد من وجود ذاكرة كافية متاحة للمنظومات الديناميكية وانتاج تجاور فى توزيع مواقع السلسلة. وفيما يلى بعض أمثلة على دالة FRE.

```
DIM NewI(Max.Min)
..
RM = FRE(-1)
..
Max = Max + 2: Min = Min + 2
REDIM NewI(Max.Min)
```

يوضح المثال السابق كيف يمكن استخدام دالة FRE فى تحديد ما إذا كانت هناك ذاكرة كافية متاحة لاعادة تحديد أبعاد المنظومة New1 أم لا.

```
..
AvMem = FRE("Junk")
..
```

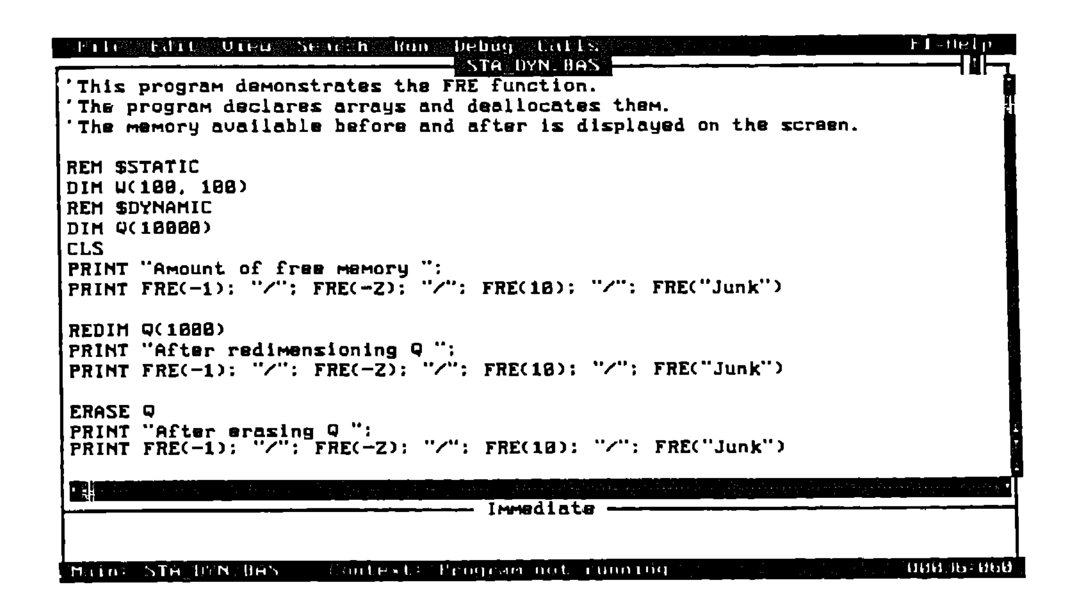
يستخدم هذا المثال دالة FRE مع سلسلة ويتسبب فى ضغط مخزن السلسلة المتاح فى مجموعة واحدة. ويساعد ذلك فى تحسين أداء البرنامج.

عملية تقليدية

تقوم فى هذه العملية بتعديل البرنامج المقدم فى الدرس المائة والسابع والثلاثين للتعامل مع دالة FRE. ابدأ بتحميل بيسك السريع.

١ – اختر فتح ملف Open وحمل البرنامج STA DYN.BAS.

٢ – عدل الملاحظات وعبارات PRINT كما هو مبين لاستخدام مؤشرات مختلفة مع دالة FRE.



```
'This program demonstrates the FRE function.
'The program declares arrays and deallocates them.
'The memory available before and after is displayed on the screen.

REM $STATIC
DIM W(100, 100)
REM $DYNAMIC
DIM Q(10000)
CLS
PRINT "Amount of free memory ":
PRINT FRE(-1): "/": FRE(-2): "/": FRE(10): "/": FRE("Junk")

REDIM Q(1000)
PRINT "After redimensioning Q ":
PRINT FRE(-1): "/": FRE(-2): "/": FRE(10): "/": FRE("Junk")

ERASE Q
PRINT "After erasing Q ":
PRINT FRE(-1): "/": FRE(-2): "/": FRE(10): "/": FRE("Junk")
```

٣ – نفذ البرنامج ولاحظ المخرجات. لاحظ أن الكل، باستثناء FRE (-1)، يعيد نفس القيمة فى هذا البرنامج. وتكون المخرجات على النحو التالى :

```
Amount of free memory  262384 / 1162 / 49808 / 49808
After redimensioning Q  298384 / 1162 / 49808 / 49808
After erasing Q  302480 / 1162 / 49808 / 49808




Press any key to continue
```

٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٥ - انتقل إلى الدرس المائة والثالث والعشرين للاستمرار فى تسلسل التعلم.

الدرس الحادى والخمسون

دالة FREEFILE

الوصف

تعطى دالة FREEFILE رقم الملف التالى المتاح فى بيسك السريع وتكوينها هو كمايلى:

```
FREEFILE
```

والرقم الذى تعيده هو رقم ملف صحيح ولم يستخدم فى بيسك السريع. ويستخدم هذا الرقم فى تحديد رقم الملف فى عبارة OPEN.

التطبيقات

دالة FREEFILE مفيدة فى انتاج أرقام ملفات صحيحة أثناء تنفيذ البرنامج. ويمكن بواسطة دالة FREEFILE أن تنتج البرامج الفرعية والأجزاء الفرعية ملفات باستخدام أرقام الملفات التى تنتج عند وقت التنفيذ وفيما يلى أمثلة لدالة FREEFILE.

مثال١

```
Input "Enter file name": FileName$
NextFile = FREEFILE
OPEN FileName$ FOR RANDOM AS NextFile
```

مثال٢

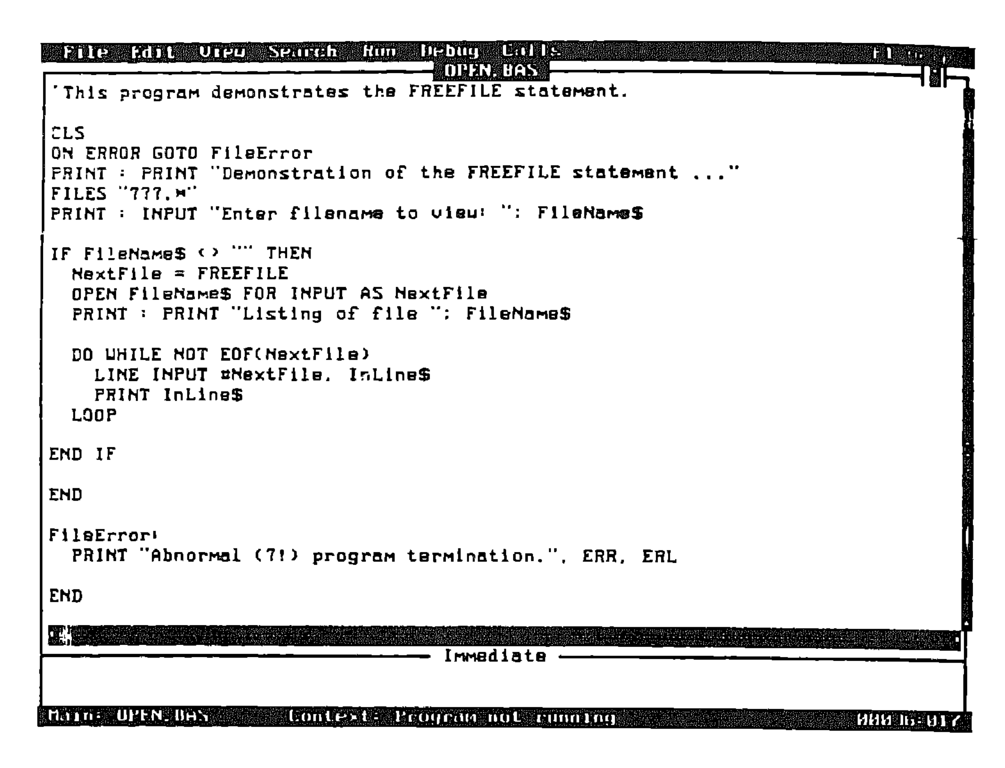
```
SUB  LoadFile(FileName AS STRING)
     NextFN = FREEFILE
     OPEN FileName FOR INPUT AS NextFN
     ..
END SUB
```

عملية تقليدية

تستخدم هذه العملية البرنامج الموجود فى الدرس الرابع والتسعين فى توضيح استخدام دالة FREEFILE. ابدأ بتحميل بيسك السريع.

١ - اختر Open من قائمة File وحمل البرنامج OPEN.BAS.

٢ - عدل عبارة OPEN وعبارة PRINT وسطر التعليق ليتوا ءما مع القائمة التالية :



```
File  Edit  View  Search  Run  Debug  Calls                                  F1
                                   OPEN.BAS
'This program demonstrates the FREEFILE statement.

CLS
ON ERROR GOTO FileError
PRINT : PRINT "Demonstration of the FREEFILE statement ..."
FILES "???.*"
PRINT : INPUT "Enter filename to view: "; FileName$

IF FileName$ <> "" THEN
  NextFile = FREEFILE
  OPEN FileName$ FOR INPUT AS NextFile
  PRINT : PRINT "Listing of file "; FileName$

  DO WHILE NOT EOF(NextFile)
    LINE INPUT #NextFile, InLine$
    PRINT InLine$
  LOOP

END IF

END

FileError:
  PRINT "Abnormal (?!) program termination.", ERR, ERL

END
                                  Immediate
Main: OPEN.BAS          Context: Program not running                  00016:017
```

٣ - نفذ البرنامج. لاحظ استخدام دالة FREEFILE فى البرنامج. لاحظ كذلك أن الشاشة تبين قائمة مختلفة من الملفات. اختر ملفاً سبق لك حفظه فى أحد الدروس السابقة. ويجب أن يكون الملف ملف ASCII لكى ينفذ البرنامج بطريقة صحيحة. وفيما يلى عينة للتنفيذ.

```
Demonstration of the FREEFILE statement ...
C:\QB
        .    <DIR>            ..  <DIR> BC       .EXE      QB       .EXE
LIB     .EXE      QB       .HLP      QB       .LIB      QB       .QLB
QB      .PIF      QB       .BI       BOX      .BAS      BOX      .OBJ
BOX     .EXE      EXE      .MAP      ASC      .BAS      ABS      .BAS
UP      .TXT      002                003                004
005               W        .DAT      SUB      .BAS      QB       .INI
STR     .BAS      A        .BAS
 2791424 Bytes free

Enter filename to view: ? ASC.BAS

Listing of file ASC.BAS
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
  Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%




Press any key to continue
```

٤ – ارجع إلى البرنامج. اختر New واختر عدم حفظ البرنامج.

٥ – انتقل إلى الدرس السادس والأربعين للاستمرار فى تسلسل التعلم.

الدرس الثانى والخمسون

عبارة FUNCTION

الوصف

توضح وتعرف عبارة FUNCTION دالة فى بيسك السريع، وتكوينها هو كما يلى :

```
FUNCTION name (parameter list) STATIC
     ..
     name = expression
END FUNCTION
```

جزء name هو اسم الدالة وهو اسم متغير لبيسك سريع صحيح ولايزيد طوله عن 40 خانة. ويمكن أن يشمل الاسم معرفاً للنوع لتحديد نوع النتيجة التى تعيدها الدالة. وجزء parameter list هو قائمة المؤشرات التى تتوقعها الدالة وتقبلها. وتتحدد محتويات القائمة بفواصل. وقائمة المؤشرات اختيارية حيث يمكن تعريف الدالة بدون أى مؤشرات. (تكوين قائمة المؤشرات موصوف فى المقطع التالى). الكلمة المحجوزة STATIC تستخدم اختيارياً لتحديد ماإذا كانت المتغيرات داخل الدالة تحتفظ بنفس قيمها بين الاستدعاءات أم لا. وجزء expression المحدد لاسم الدالة هو القيمة المراد تحديدها، وتعود كنتيجة للدالة. تحدد عبارة END FUNCTION انتهاء جسم الدالة.

وقائمة المؤشرات لها التكوين التالى :

```
var1 AS type, var2 AS type,...
```

ويحدد جزء A Stype نوع بيانات المتغيرات. وبديلا لذلك عندما يكون المتغير عبارة عن منظومة فيوضع جزء من القائمة على هيئة قوسين فارغين كما يلى :

```
var1(), var2(),...
```

لاحظ غياب مواصفة A Stype.

التطبيقات

عبارة FUNCTION هى طريقة أخرى لعزل قطع من الشفرة أجرى عليها اختبار وتنفذ نشاطاً معرفاً تعريفاً جيداً. كل المتغيرات فى FUNCTION تكون محلية ويوضع لها قيم ابتدائية

مساوية اصفاراً أو فراغات وذلك قبل تنفيذ الدالة. وتقدم الكلمة الرئيسية الاختيارية STATIC آلية لحفظ قيم المتغيرات المحلية بين استدعاءات الدالة كما هى. وحيث إن الدالة تعيد نتيجة واسم الدالة هو اسم متغير بيسك سريع فيستخدم اسم الدالة فى عبارات التحديد. ويمكن للدالة المعرفة بعبارة FUNCTION ان تكون لها خاصية الاعادة الذاتية أى إنها تستدعى نفسها. وعندما تتسم الدالة بأنها لها خاصية الاعادة الذاتية فلايكون استخدام الكلمة المحجوزة STATIC فكرة طيبة مع الدالة. وفيما يلى بعض أمثلة لعبارات FUNCTION.

مثال١

```
FUNCTION  ConstStr$(Ch$ AS STRING)
     ..
     ..
END FUNCTION
```

مثال٢

```
FUNCTION  RandStr$(Range$ AS STRING)
     IF Range$ = "" THEN EXIT FUNCTION
     ..
END FUNCTION
```

مثال٣

```
FUNCTION  KeepCount(T1()) STATIC
     ChCnt = ChCnt + 1
     ..
END FUNCTION
```

مثال٤

```
FUNCTION Factorial(Num AS INTEGER)
     ..
     IF Num > 1 THEN Factorial = Num * Factorial(Num - 1)
     ..
END FUNCTION
```

يبين المثال الأخير كيف تتسم الدالة بخاصية الاعادة الذاتية.

عملية تقليدية

تستخدم هذه العملية البرنامج المقدم فى الدرس السادس والثمانين. يقدم البرنامج توضيحاً جيداً لعبارة FUNCTION. ابدأ بتحميل بيسك السريع.

١ - اختر Open وحمل البرنامج LRTRIM.BAS.

٢ – نفذ البرنامج ولاحظ استخدام عبارة FUNCTION فى البرنامج.

٣ – اختر New مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٤ – انتقل إلى الدرس المائة والسادس والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الثالث وخمسون

عبارتا GET و PUT

الوصف

عبارة GET : تقرأ عبارة GET بيانات من ملف فى متغير أو ذاكرة احتياطية للاتصال العشوائى. وتكوينها هو كما يلى :

```
GET #filenum, recordnum, variable
```

جزء filenum هو رقم الملف المحدد للملف في عبارة OPEN. ويعطى جزء recordnum موضع بداية السجل لبدء قراءة البيانات. وفى ملفات الاتصال العشوائى يكون recordnum هو رقم السجل الفعلى. وفى الملفات الثنائية يكون رقم السجل هو موضع البايت الذى تبدأ القراءة عنده. وعندما يحذف هذا الجزء فتقرأ البيانات بدءاً بموقع انتهاء آخر سجل سبق قراءته. وأقصى رقم سجل هو 2,147,483,647. وجزء variable يتلقى البيانات المقروءة. وعند استخدام هذا الجزء فلايكون ضرورياً استخدام دوال تحويل الحقل (CVS, CVD, CVL, CVI) ويجب عدم استخدام عبارة FIELD مع الملف المستخدم وذلك لأن عبارة FIELD تنظم البيانات بطريقة مختلفة عن عبارة GET. ومع وجود الملفات فى الصيغة الثنائية فيمكن استخدام أى متغير وتقرأ عبارة GET عدد البايت كلها التى يسمح بها المتغير. وعندما تستخدم سلسلة متغيرة الطول كمقصد فتقرأ دالة GET عدداً من البايت يناظر ما يسمح به طول السلسلة الحالية. وجزئى re- cordnum و variable اختياريان.

عبارة PUT : تكتب عبارة PUT فى ملف وذلك من متغير أو ذاكرة احتياطية للاتصال المباشر. وتكوينها هو كما يلى :

```
PUT #f'.ienum, recordnum, variable
```

والأجزاء شبيهة بأجزاء عبارة GET فيما عدى أن الغرض هو كتابة ملف بدلاً من القراءة من ملف. عند استخدام جزء variable فلايكون ضرورياً استخدام دوال تحويل البيانات ,MKD$) MKS$, MKI$, MKL$) ويجب الا تستخدم عبارة FIELD مع الملف لنفس السبب المذكور مع عبارة GET.

التطبيقات

تسمح عبارات GET و PUT بطرق أكثر مرونة للاتصال بملفات اتصال عشوائى. فنسمح بالاتصال بملفات على المستوى الثنائى. ومع تقديمها المرونة فإنها تتطلب كذلك كما أكبر من المسئولية من قبل المترجم. وفيما يلى أمثلة عليها :

مثال١

```
OPEN "Config.Dat" FOR RANDOM AS #1 LEN = 80
..
INPUT "Monochrome or Color ": ScrType$
..
PUT #1,,ScrType$
```

مثال٢

```
TYPE  NewDat
     F1 AS STRING * 20
     F2 AS STRING * 20
END TYPE
DIM DatRec AS NewDat
OPEN "New.Lst" FOR RANDOM AS #4 LEN = 40
FIELD #4 20 AS Field1, 20 AS Field2
..
GET #4
..
LINE INPUT #4, DatRec
..
```

يوضح مثال ٢ كيفية استخدام عبارة FIELD مع عبارة GET. لاحظ بصفة خاصة أن عبارة GET تستخدم بدون متغير. وتستخدم عبارة LINE INPUT# فى نقل بيانات من الذاكرة الاحتياطية لملف عشوائى إلى المتغير DatRec. كما يمكن استخدام عبارة INPUT# كذلك.

عندما تستخدم عبارات GET و PUT فى الاتصالات مع سجلات ثابتة الطول فتنتظر العبارات بلا نهاية لرموز رقم السجل. استخدم هذه السمة بحذر.

وتكتب عبارات GET و PUT كما يلى عند ترك جزء أو أكثر من جزء بدون استخدام :

مثال١

```
GET #1,,InputStr$      'the recordnum is omitted
PUT #1,,InputStr$
```

مثال٢

```
GET #2,2              'the variable is omitted
PUT #2,2
```

مثال٣

```
GET #1          'the recordnum and variable are omitted
PUT #1
```

عملية تقليدية

توضح هذه العملية استخدام عبارات GET و PUT. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This program demonstrates the use of GET and PUT statements. The program
'creates a file using the PUT statement and reads it back using the GET
'statement.
TYPE CustType
   CustName AS STRING * 25
   CustNum AS INTEGER
   CustType AS STRING * 2
END TYPE

DIM Customer AS CustType
OPEN "Cust.Fil" FOR RANDOM AS #2
RecCnt = 1
CLS
DO WHILE UCASE$(Choice$) <> "Y"
   INPUT "Enter Customer name: "; Customer.CustName
   INPUT "Enter Customer number: "; Customer.CustNum
   INPUT "Enter Customer type (A,B,C): "; Customer.CustType
   INPUT "Done ? (Y/N): "; Choice$
   PUT #2, RecCnt, Customer
   RecCnt = RecCnt + 1
LOOP
CLOSE #2

PRINT "Press any key to continue ..": INPUT t$
OPEN "Cust.Fil" FOR RANDOM AS #2
RecCnt = 1
GET #2, 1, Customer

DO WHILE NOT EOF(2)
   PRINT RecCnt; Customer.CustName, Customer.CustNum, Customer.CustType
   RecCnt = RecCnt + 1
   GET #2, RecCnt, Customer
LOOP

                         Immediate
Main  <Untitled>   Context: Program not running
```

٢ – نفذ البرنامج. اكتب البيانات التالية واضغط على مفتاح الادخال بعد ادخال كل مدخل. بعد آخر ملقن اضغط على أى مفتاح. لاحظ استخدام عبارات GET و PUT فى البرنامج.

```
Enter Customer name: ? Microsoft Inc.
Enter Customer number: ? 333
Enter Customer type (A,B,C): ? A
Done ? (Y/N): ? n
Enter Customer name: ? More Money Corp.
Enter Customer number: ? 1222
Enter Customer type (A,B,C): ? C
Done ? (Y/N): ? n
Enter Customer name: ? Singapore Scents Co.
Enter Customer number: ? 999
Enter Customer type (A,B,C): ? B
Done ? (Y/N): ? Y
Press any key to continue ..
?
 1 Microsoft Inc.                          333         A
 2 More Money Corp.                        1222        C
 3 Singapore Scents Co.                    999         B






Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الثامن والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الرابع والخمسون

عبارات GET GRAPHICS و PUT GRAPHICS

الوصف

تتعامل عبارات GET و PUT المذكورة فى هذا الدرس مع صور الرسومات وتنسخ عبارة GET نسخاً من صور الرسومات من الشاشة أما عبارة PUT فتعيد صور الرسومات المخزنة بواسطة عبارة GET إلى الشاشة. وتكوين العبارتين هو كما يلى :

```
GET STEP (x1,y1)-STEP(x2,y2),array
PUT STEP(x,y),array,action
```

عبارة GET : جزء STEP اختيارى ويحدد أن الاحداثيات نسبية إلى آخر نقطة يشار اليها. ولايستخدم هذا الجزء مرتين فى نفس عبارة GET سواء كانت احداثيات البداية نسبية أو كانت احداثيات النهاية نسبية أو كان كل من الاحداثيات للبداية والنهاية مطلقة. تشير أجزاء (x1, y1) و (x2, y2) إلى احداثيات البداية والنهاية لمنطقة مستطيلة يراد نسخها. وجزء array هو الموقع الذى تنسخ فيه منطقة الشاشة ويكون من النوع الصحيح. ويتحدد حجم المنظومة باستخدام الصيغة التالية :

```
Size = 4 + INT(((x2 - x1 - 1) *
       (bits per pixel per plane) + 7) / 8) *
       planes *(y2 - y1) + 1
```

تعتمد قيم المستويات وعدد البت فى كل نقطة رسم واحدة فى المستوى الواحد على الشاشة. ويعطى الجدول التالى البت لكل نقطة رسم لكل مستوى والمستويات لحالات الشاشة المختلفة.

الحالة	بت لكل نقطة رسم لكل مستوى	المستويات
1	2	1
2	1	1
7	1	4

الحالـة	بت لكل نقطة رسم لكل مستوى	المستويات
8	1	4
9	1	(EGA لذاكرة 64K)2 (EGA أكبر من 64K لذاكرة)4
10	1	2
11	1	1
12	1	4
13	8	1

عبارة PUT : ينفذ جزء STEP نفس الوظيفة فى عبارة PUT مثلما يفعل فى حالة عبارة GET. ويعطى جزء (x,y) احداثيات البداية المراد وضع الهدف عندها. جزء array هو منظومة تحتوى على الهدف المحفوظ بواسطة عبارة GET. وجزء action يحدد كيفية وضع الصورة على الشاشة فوق الأشياء الموجودة. والاجراءات actions هى PSET و PRESET و AND و OR و XOR. ويصف الجدول التالى تقاطع الأشياء المراد وضعها مع أشياء الشاشة لكل من أفعال الاجراءات المسموح بها.

الفعـل	الوصـف
PSET	ينقل الشىء نقطة بنقطة وتكون الألوان هى نفسها مثل الألوان المراد نقلها.
PRESET	مثل PSET فيما عدى وضع سالب للصورة.
AND	تتسبب فى عملية اضافة AND على الشاشة والشىء المراد وضعه. وتظل الألوان المتشابهة فى الشىء والشاشة كما هى. أما الألوان غير المتشابهة فتظل غير متشابهة.
OR	تتسبب فى عملية أو OR على الشاشة والشىء المراد وضعه.
XOR	تتسبب فى عملية XOR على الشاشة والشىء المراد وضعه. يسمح بوضع الشىء المراد وضعه بدون حذف الصورة الموجودة.

تتسبب مؤثرات بوليان AND و OR و XOR فى معالجة عبارة GET بنفس الطريقة التى تعالج بها المؤثرات المنطقية الأعداد.

التطبيقات

عبارات GET و PUT هى أكثر نفعاً فى الرسوم المتحركة فيمكن استخدامها كذلك فى أساليب عمل نوافذ الظهور pop-up فى بيئة اعداد الرسومات. وفيما يلى بعض أمثلة لعبارات GET و PUT.

```
SCREEN 1
'$DYNAMIC
ArrSize = 1024
..
REDIM ObjArr(ArrSize)
..
GET (UlX,Uly)-(Lrx,Lry),ObjArr
..
PUT (Ulx,Uly),ObjArr,PSET

FUNCTION ASize(Ulx,Uly,Lrx,Lry)
     ASize = 4+INT(((Lrx-Ulx+1)*1+7)/8)*1*(Lry-Uly+1)
END FUNCTION
```

توضح الدالة FUNCTION ASize دالة حساب حجم حالة الشاشة المعطاة 2.

عملية تقليدية

هذه العملية هى توضيح بسيط لعبارات GET و PUT المستخدمة مع الرسومات. اكمل فقط إذا كانت لديك امكانيات رسومات ملونة تدعم ذلك.

ابدأ بتحميل بيسك السريع.

١ – حمل البرنامج DRAW.BAS وعدل البرنامج كما هو مبين فى القائمة التالية.

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              DRAW.BAS
'This program demonstrates the use of the GET and PUT statements.

'Declare arrays as dynamic
'$DYNAMIC

'Draw the face on the screen
SCREEN 1
DRAW "c1 u5 r5 d5 l5 "
DRAW "bm+20,0 u5 r5 d5 l5 "
DRAW "Bm-8,0 d10"
DRAW "bm-5,5 r10"
DRAW "bm+15,0 u30 l35"
xp = POINT(0): yp = POINT(1)
DRAW "d30 f10 r15 e10"
DRAW "bm=" + VARPTR$(xp) + ",=" + VARPTR$(yp)

FOR Cnt = 1 TO 12
   DRAW "c2 ne5 "
   DRAW "bm+3,0"
NEXT

'Calculate array size
asize = 4 + INT((65 * 2 + 7) / 8) * 1 * 65
asize = asize / 2
'Redimension the array, and create a second array to hold a piece of the
'blank screen
REDIM ObjArr(asize) AS INTEGER, ObjBlank(asize) AS INTEGER

'Get the object
GET (143, 70)-(158, 125), ObjArr
'Get a piece of the blank screen
GET (1, 1)-(65, 65), ObjBlank

'Put the blank piece on the object
PUT (143, 70), ObjBlank, AND
'Put the object at the top left corner of the screen
PUT (1, 1), ObjArr%

                              Immediate

Main: DRAW.BAS          Context: Program not running          00054:001
```

٢ – نفذ البرنامج ولاحظ استخدام عبارات GET و PUT المستخدمة مع الرسومات في البرنامج.

٣ – ارجع إلى البرنامج واحفظ البرنامج كملف نصي تحت الاسم GPGRAPH.BAS مع اخلاء الشاشة.

٤ – انتقل إلى الدرس الرابع والثمانين للاستمرار فى تسلسل التعلم.

الدرس الخامس والخمسون

تكوين GOSUB.. RETURN

الوصف

يستخدم التكوين GOSUB.. RETURN فى النداء على برنامج فرعى والعودة منه. ولايشمل ذلك برامج فرعية سبق توضيحها باستخدام عبارات SUB أو FUNCTION. وتكوينه هو كما يلى :

```
GOSUB line number or line label
RETURN line number or line label
```

الجزء	الوصف
GOSUB	كلمة من كلمات بيسك المحجوزة تعمل تفريعاً إلى برنامج فرعى بدءاً برقم سطر معين أو باسم سطر معين.
line number or line label	رقم السطر أو اسم السطر الذى يتم التفريغ إليه.
RETURN	كلمة من كلمات بيسك المحجوزة والتى تعيد التحكم إلى العبارة الموجودة عند رقم سطر معين أو اسم سطر معين على مستوى الجزء (البرنامج) فقط. فإذا لم يكن محدداً رقماً أو اسماً للسطر فيعود البرنامج إلى العبارة التى تلى عبارة GOSUB.

يمكن استدعاء البرامج الفرعية أى عدد من المرات، كما يمكن استدعاؤها كذلك من برامج فرعية أخرى وداخل نفسها. والقيد الوحيد على عدد البرامج الفرعية التى يمكن أن تتداخل مع بعضها البعض هو مكان الرصة المتاح.

التطبيقات

البرامج الفرعية عبارة عن طريقة جيدة لعزل قطع من الشفرة يتكرر استخدامها فى أجزاء مختلفة من البرامج. ويساعد مثل هذا العزل على إعداد برمجة فعالة وعلى تسهيل قراءة البرنامج

وتسهيل صيانته كذلك. وتكوين GOSUB.. RETURN هو طريقة فعالة لاستدعاء برامج فرعية غير موضحة باستخدام عبارات SUB أو FUNCTION. وفيما يلى بعض الأمثلة :

مثال١

```
    GOSUB 200
    ..
200 PRINT "The result is " Qrt%
    ..
    RETURN
```

هذا مثال لاستدعاء بسيط لبرنامج فرعى باستخدام GOSUB.. RETURN.

مثال٢

```
GOSUB 1500: GOSUB 2000: GOSUB CheckList
```

هذا مثال لكيفية كتابة شفرة أكثر من عبارة GOSUB واحدة فى نفس السطر. وحيث إن عبارة RETURN تعود إلى عبارة البرنامج التى تلى GOSUB فتستدعى البرامج الفرعية واحداً تلو الآخر.

مثال٣

```
    GOSUB MoveData
    ..
ContMove:
    ..
MoveData:
    ..
    RETURN ContMove
```

هذا مثال لكيفية عودة عبارة RETURN إلى عبارة من عبارات البرنامج غير العبارة التى تلى GOSUB مباشرة. وفى هذه الحالة تمرر RETURN التحكم إلى الاسم ContMove.

مثال٤

```
    GOSUB  Truncate
    ..
Psl:
Truncate:
    ..
    GOSUB  StripBlanks: GOSUB Tokenize
    ..
Tsl:
    RETURN Psl
StripBlanks:
    ..
    RETURN Tsl
Tokenize:
    ..
    RETURN
```

يبين هذا المثال كيفية تداخل البرامج الفرعية في بعضها البعض فالبرنامج الفرعى الذى يبدأ عند الاسم Truncate يستدعى البرنامجين الفرعيين الموجودين عند الاسمين StripBlanks

و Tokenize. من الممكن عمل شفرة لعبارات GOSUB على مستوى StripBlanks و Token-ize كذلك. لاحظ أن الاسم المستخدم فى عبارة RETURN الموجودة فى Truncate يشير إلى Ps1 والاسم المستخدم فى عبارة RETURN الموجودة فى StripBlanks يشير إلى Ts1. وهذا بسبب أنه ليس من الممكن العودة إلى مدى أكبر من المدى الذى استدعى البرنامج الفرعى. وهذا يعنى أن البرنامج لايستطيع العودة إلى Ps1 من StripBlanks ويترك Truncate حيث إنه لم يستدع من هذا المستوى.

مثال٥

```
        GOSUB  Factorial
        ..
Factorial:
        ..
        IF (f1 <> f2) THEN
           f1 := f1 - 1
           GOSUB Factorial
        ELSE RETURN
        END IF
```

هذا مثال لكيفية استدعاء برنامج فرعى لنفسه. فالبرنامج الفرعى الموجود عند الاسم Facto-rial يستدعى نفسه حيث يتم حساب المضروب. ويعرف هذا بالإعادة الذاتية recursion. هذا النوع من الشفرة يجب أن يستخدم بحذر شديد لتجنب إثارة مكان الرصة وحدوث انهيار للبرنامج.

عملية تقليدية

هذا المثال يوضح استخدام تكوين GOSUB.. RETURN. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                                F1=Help
                              <Untitled>
CLS
Sel1$ = "Selection 1": Sel2$ = "Selection 2": Sel3$ = "Selection 3"
Sel4$ = "Quit": Sel5$ = "Enter the number, or Q to quit"
LOCATE 2, 30: PRINT "MENU SELECTION"
LOCATE 4, 30: PRINT Sel1$
LOCATE 5, 30: PRINT Sel2$
LOCATE 6, 30: PRINT Sel3$
LOCATE 7, 30: PRINT Sel4$
LOCATE 8, 30: PRINT Sel5$
Y = 4: X = 30
GOSUB GetKey

DO WHILE ch$ < > "Q"
  IF ch$ = "1" THEN
     GOSUB Select1
  ELSEIF ch$ = "2" THEN GOSUB Select2
  ELSEIF ch$ = "3" THEN GOSUB Select3
  END IF
  GOSUB GetKey
LOOP

END

GetKey:
  ch$ = ""
  DO WHILE ch$ = ""
    ch$ = INKEY$
  LOOP
  ch$ = UCASE$(ch$)
RETURN

Select1:
  LOCATE 10, 30: PRINT "Selection One  "

RETURN

Select2:
  LOCATE 10, 30: PRINT "Selection Two  "

RETURN

Select3:
  LOCATE 10, 30: PRINT "Selection Three"

RETURN

                              Immediate
```

٢ – اضغط على Shift-F5 للتنفيذ. لاحظ كيفية عمل GOSUB.. RETURN فى البرنامج. اكتب Q لإنهاء البرنامج.

```
MENU SELECTION

Selection 1
Selection 2
Selection 3
Quit
Enter the number, or Q to quit

Selection Three
```

٣ – اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ – اختر Save من قائمة File واكتب GOSUB.BAS كاسم للملف. حدد الملف من النوع النصى واحفظ هذا البرنامج.

٥ – اختر New من قائمة File لإخلاء الشاشة.

٦ – انتقل إلى الدرس السادس والثلاثين للاستمرار فى تسلسل التعلم.

الدرس السادس والخمسون

عبارة GOTO

الوصف

عبارة GOTO هى تفريع غير شرطى، وهذا يعنى أن البرنامج يستمر على أى حال من الأحوال مع السطر أو الاسم الذى تشير إليه عبارة GOTO. ويمكن أن يكون التفريع غير الشرطى على نفس مستوى عبارة GOTO فقط. ولايمكنك أن تستخدم GOTO فى الدخول فى برنامج فرعى أو دالة أو الخروج منها يكون معرفاً (أو معرفة) بواسطة SUB أو FUNCTION أو DEF FN. وتكوينها هو كما يلى :

```
GOTO line number|line label
```

الجـزء	الوصـــف
GOTO	كلمة من كلمات بيسك السريع المحجوزة.
linenum 1 line label	رقم صحيح لسطر، يتراوح من 0 إلى 65,529، أو اسم صحيح لسطر يبدأ بحرف. وهذا هو مقصد أمر التفريع.

التطبيقات

الاستخدامات مفيدة إلا أن الاستخدامات الكثيرة وغير المميزة لعبارة GOTO يمكن أن تقود إلى شفرة تشبه المكرونة الاسباكيتى (صعبة التصحيح وصعبة القراءة والفهم وصعبة الصيانة). ويوصى بشدة بالتحكم فى مسار البرنامج باستخدام العبارات IF.. THEN.. ELSE و DO.. LOOP و FOR.. NEXT و WHILE.. WEND و SELECT.. CASE وذلك بدلاً من استخدام عبارات GOTO.

```
GOTO ErrorRoutine
GOTO 233
GOTO Terminate
```

عملية تقليدية

يوضح هذا القسم استخدام عبارات GOTO. استمر على النحو التالى :

١ - إذا كنت تستمر من درس إلى آخر فعليك باخلاء منقح بيسك السريع وذلك باختيار New من قائمة File مع اختيار عدم حفظ البرنامج المحمل حالياً وإلا فابدأ بيسك السريع من البداية.

٢ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the GOTO statement.
CLS
INPUT "Enter a value ..": v
IF v < 10 THEN GOTO LessNTen
IF v < 20 AND v > 10 THEN GOTO LessNTwenty
IF v > 20 AND v < 30 THEN GOTO LessNThirty
PRINT "Value more than 30.": END
LessNTen: PRINT "Value is less than 10.": END
LessNTwenty: PRINT "Value is more than 10 and less than 20.": END
LessNThirty: PRINT "Value is more than 20 and less than 30.": END

                          Immediate

Main: <Untitled>     Context: Program not running              00010:006
```

٣ - اضغط على Shift-F5 لتنفيذ البرنامج.

```
Enter a value ..? 1
Value is less than 10.



Press any key to continue
```

٤ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج. اضغط على Alt-F واكتب F لحفظ هذا البرنامج.

٥ - اكتب GOTO واضغط على Tab واستخدم مفاتيح الأسهم لتحديد أن الملف نصى. اضغط على Tab وعلى مفتاح الإدخال لحفظ هذا البرنامج. تذكر أن بيسك السريع يضيف التوسع لاسم الملف BAS.

٦ - انتقل إلى الدرس السبعين للاستمرار فى تسلسل التعلم.

الدرس السابع والخمسون

دالـة HEX$

الوصف

تعيد دالة HEX$ القيمة السادسة عشرية لتعبير عددى عشرى، وتكوينها هو كما يلى :

```
HEX$(numeric expression)
```

يقرب التعبير العددى إلى أقرب قيمة صحيحة. فإذا وقعت القيمة خارج المدى الصحيح (من 0 إلى 32767) فإنه يتحول إلى قيمة صحيحة طويلة فى الصورة السادسة عشرية. والقيمة التى تعود تكون سلسلة ولايمكن استخدامها على هذا فى الحسابات.

التطبيقات

تستخدم دالة HEX$ عندما تكون هناك حاجة إلى معرفة القيمة السادسة عشرة لتعبير عددى عشرى. ومثل هذا التحويل مفيد للمبرمجين الذين يتعاملون مع مثل هذا التمثيل للبيانات. وفيما يلى بعض أمثلة لدالة HEX$.

```
PRINT HEX$(4096)
H$ = HEX$(128*64)
T$ = HEX$(12/3)
T$ = HEX$(1.09*13)
```

عملية تقليدية

دالة HEX$ موضحة فى العملية التالية. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                        F1=Help
                                 <Untitled>
CLS
FOR A% = 33 TO 128
  AHex% = ASC(CHR$(A%))
  PRINT CHR$(A%); " "; HEX$(AHex%); "/ ";
NEXT

                                 Immediate

 Main: <Untitled>      Context: Program not running                 00004:033
```

٢ – نفذ البرنامج. لاحظ أن البرنامج يطبع رموزاً مقروءة من جدول ASCII (من 33 إلى 128) والمكافىء السادس عشرى لقيمها ASCII.

```
! 21/ " 22/ # 23/ $ 24/ % 25/ & 26/ ' 27/ ( 28/ ) 29/ * 2A/ + 2B/ , 2C/ - 2D/ .
2E/ / 2F/ 0 30/ 1 31/ 2 32/ 3 33/ 4 34/ 5 35/ 6 36/ 7 37/ 8 38/ 9 39/ : 3A/ ; 3B
/ < 3C/ = 3D/ > 3E/ ? 3F/ @ 40/ A 41/ B 42/ C 43/ D 44/ E 45/ F 46/ G 47/ H 48/
I 49/ J 4A/ K 4B/ L 4C/ M 4D/ N 4E/ O 4F/ P 50/ Q 51/ R 52/ S 53/ T 54/ U 55/ V
56/ W 57/ X 58/ Y 59/ Z 5A/ [ 5B/ \ 5C/ ] 5D/ ^ 5E/ _ 5F/ ` 60/ a 61/ b 62/ c 63
/ d 64/ e 65/ f 66/ g 67/ h 68/ i 69/ j 6A/ k 6B/ l 6C/ m 6D/ n 6E/ o 6F/ p 70/
q 71/ r 72/ s 73/ t 74/ u 75/ v 76/ w 77/ x 78/ y 79/ z 7A/ { 7B/ | 7C/ } 7D/ ~
7E/ ⌂ 7F/ Ç 80/




Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الحادى والتسعون للاستمرار فى تسلسل التعلم.

الدرس الثامن والخمسون

عبارة IF..THEN..ELSE

الوصف

تضع معظم البرامج شروطاً تحتاج لأن تتحقق قبل أن يمكن تنفيذ جزء معين من الشفرة. ويعرف هذا بأنه تفريع شرطى conditional branching. ويتحقق ذلك فى بيسك السريع باستخدام عبارات IF..THEN..ELSE. وهناك طريقتان لكتابة عبارة IF..THEN..ELSE.

الطريقة الأولى تستخدم تكوين سطر واحد له الشكل التالى :

```
IF boolean expression THEN statement ELSE statement
```

الجـزء	الوصـف
IF	كلمة من كلمات بيسك السريع المحجوزة.
boolean expression	تعبير تختبر نتيجته لأن تكون صحيح أو خطأ. تنفذ العبارات التى تأتى بعد THEN إذا كانت نتيجة التعبير صحيح أما إذا كانت النتيجة خطأ فتنفذ العبارات التى تلى ELSE.
THEN	كلمة من كلمات بيسك السريع المحجوزة.
statement	عبارة من عبارات بيسك السريع.
ELSE	كلمة من كلمات بيسك السريع المحجوزة.
statement	عبارة من عبارات بيسك السريع.

تستخدم الطريقة الثانية تكوين مجموعة بالشكل التالى :

```
IF boolean expression THEN
   statement block 1
ELSEIF boolean expression THEN
   statement block 2
ELSE
   statement block 3
END IF
```

والأجزاء الجديدة فى الطريقة الثانية موصوفة على النحو التالى :

الجــزء	الوصــف
statement block1, 2, and 3	عبارات برنامج بيسك السريع التى يراد تنفيذها عندما يتحقق الشرط.
ELSE IF	كلمـة من كلمـات بيـسك السـريع المحـجوزة. تستـخـدم فى التداخل فى عبارة IF..THEN.
END IF	كلمـة من كلمات بيسك السريع المحجوزة. وهى مطلوبة لانهاء تكوين IF..THEN فى الطريقة الثانية.

فى الطريقة الأولى يمكن أن تحتوى عبارات بيسك التى تنفذ بعد THEN أو ELSE على تعليمات تفريعات اضافية مثل GOTO و GOSUB. فاذا مـا أشار التفريـع إلى رقم سطر فتكون GOTO اختيارية. وإذا ما اشار التفريع إلى اسم سطر فتكون الكلمات المحجوزة GOTO أو GOSUB لازمة مع الاسم.

وفى كل من الطريقتين يكون جزء ELSE من التكوين اختيارياً. فإذا لم يكن الجزء ELSE موجوداً فيستمر البرنامج فى التنفيذ من عند العبارة التالية.

التطبيقات

تكوين IF..THEN..ELSE عبارة عن وسيلة جيدة جداً فى أى برنامج يأخذ قرارات مبنية على شروط تحدث أثناء تنفيذ البرنامج. الاستخدام الحكيم لتكوينات التفريع المتاحة مع طريقة المجموعة فى بيسك السريع يجعل البرامج ممتعة فى كتابتها وقراءتها وصيانتها. وفيما يلى بعض الأمثلة :

الطريقة الأولى:

```
IF  (Hours% > RegHrs%) THEN  GOTO 800 ELSE  1800
IF  (Hours% > RegHrs%) THEN  OverTime ELSE RegTime
IF (Q = SValue) THEN Q=Q^2:SValue=SQR(SValue):PP=SValue+Q: ELSE 200
```

الطريقة الثانية:

```
IF (OutStanding! <= CreditLimit!) THEN
   GOSUB AccountsUpdate
   GOSUB LedgerEntry
   GOSUB JE
ELSE
   GOSUB OverDueAccts
   GOSUB PayableAging
END IF

IF Switch = 1 THEN
   GOSUB  Step1: GOSUB Step23
   CALL Evaluate
ELSEIF Switch = 2 THEN
        GOSUB Step2: GOSUB Step24
        CALL Evaluate
        ELSEIF Switch = 3 THEN
               GOSUB Step3: GOSUB Step25
        END IF
     END IF
END IF
```

عملية تقليدية

يوضح البرنامج التالى استخدام عبارة IF..THEN..ELSE. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
CLS
PRINT "This is a guessing game."
PRINT "I have a number in mind.  You try to guess what it is."
RANDOMIZE TIMER
Guess% = INT(10 - 1) * RND + 1
PRINT "You have three guesses..."
Cnt = 1
DO WHILE (Cnt <= 3)
   INPUT "Your guess ", InG%
   IF InG% > Guess% THEN PRINT "Too high.."
   IF InG% < Guess% THEN PRINT "Too low.."
   IF InG% = Guess% THEN PRINT "Bingo !": Cnt = 4
   Cnt = Cnt + 1
LOOP
PRINT "My number was "; Guess%; " and your number was "; InG%

                          Immediate

Main: <Untitled>      Context: Program not running              00016:001
```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج وادخل ثلاثة تخمينات.

```
This is a guessing game.
I have a number in mind.  You try to guess what it is.
You have three guesses...
Your guess 4
Too low..
Your guess 9
Too high..
Your guess 5
Too low..
My number was  7  and your number was  5



Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ – من قائمة File اختر Save واكتب IFTHEN.BAS حدد شكل الملف أنه من النوع النصى واحفظ هذا البرنامج.

٥ – اختر New من قائمة File مع اخلاء الشاشة.

٦ – انتقل إلى الدرس المائة والسادس والعشرين للاستمرار فى تسلسل التعلم.

الدرس التاسع والخمسون

عبارة $INCLUDE

الوصف

عبارة INCLUDE$ تسمى بشبيه الأمر مثل STATIC$ و DYNAMIC$. ويسمح شبيه الأمر INCLUDE$ بتشغيل ملفات مصدر أخرى أثناء الترجمة. وتكوين العبارة هو كما يلى :

```
REM $INCLUDE: 'filename'
```

ومثل كل اشباه الأوامر يظهر أمر INCLUDE$ فى عبارة REM. جزء filename هو الملف المراد تشغيله أثناء الترجمة ويمكن أن يشمل اسم مسار. وبعد انتهاء تشغيل الملف (القراءة والترجمة) تستمر الترجمة بالعبارة التى تلى أمر INCLUDE$ مباشرة. والقيود الموجودة على الملفات المشمولة هى كما يلى :

- يجب الا تحتوى الملفات على عبارات SUB أو عبارات FUNCTION.
- يجب أن تحفظ الملفات التى سبق انتاجها باستخدام صيغ بيسك القديمة عن طريق خيار A أى كملف ASCII.

التطبيقات

يستخدم شبيه الأمر INCLUDE$ فى احتواء ملف خارجى أثناء الترجمة مثل توضيحات ملفات SUB و FUNCTION. وفيما يلى أمثلة لشبيه الأمر INCLUDE$:

```
REM $INCLUDE: 'Declare1.bi'
'$INCLUDE: 'Absolute.Bi'
```

عملية تقليدية

توضح هذه العملية شبيه الأمر INCLUDE$ عن طريق استخدامه فى تحميل ملف توضيحات. إذا كانت لديك أى مشاكل مع أى من هذه الخطوات ارجع إلى الملحق B (اقطع والصق، استخدام صناديق الحوار، قائمة أوامر التنقيح). ابدأ بتحميل بيسك السريع.

١ - اختر Open من قائمة File وحمل البرنامج SUB.BAS.

٢ – انقل نقطة البداية إلى عبارة DECLARE واضغط على مفتاح السهم السفلى Shift- واضغط على Shift-Del. يحذف ذلك السطر مع وضعه فى لوحة القص.

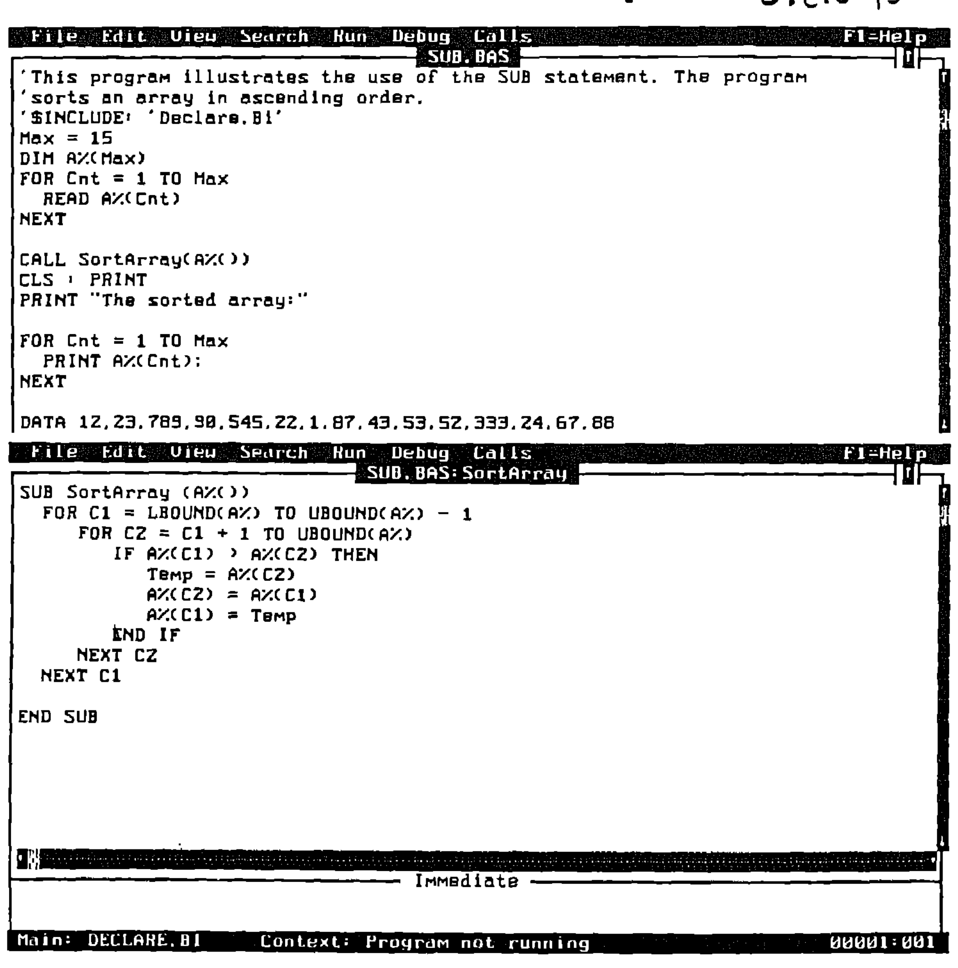
٣ – اختر Create من قائمة File واكتب اسم الملف DECLARE. BI وحدد نوع الملف بأنه In- clude واضغط على مفتاح الادخال. اضغط على Shift-Ins والصق السطر الموجود فى لوحة القص.

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              DECLARE.BI
DECLARE SUB SortArray (A%())

                              Immediate

Main: DECLARE.BI      Context: Program not running           00001:001
```

٤ – اضغط على F2 واختر SUB.BAS لتنقيحه واضف عبارة $INCLUDE$ مع أى اسم مسار لازم. ارجع إلى القائمة التالية :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              SUB.BAS
'This program illustrates the use of the SUB statement. The program
'sorts an array in ascending order.
'$INCLUDE: 'Declare.Bi'
Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
  READ A%(Cnt)
NEXT

CALL SortArray(A%())
CLS : PRINT
PRINT "The sorted array:"

FOR Cnt = 1 TO Max
  PRINT A%(Cnt);
NEXT

DATA 12,23,789,98,545,22,1,87,43,53,52,333,24,67,88
```

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         SUB.BAS:SortArray
SUB SortArray (A%())
  FOR C1 = LBOUND(A%) TO UBOUND(A%) - 1
     FOR C2 = C1 + 1 TO UBOUND(A%)
        IF A%(C1) > A%(C2) THEN
           Temp = A%(C2)
           A%(C2) = A%(C1)
           A%(C1) = Temp
        END IF
     NEXT C2
  NEXT C1

END SUB

                              Immediate

Main: DECLARE.BI      Context: Program not running           00001:001
```

٥ – نفذ البرنامج ولاحظ تأثير عبارة $INCLUDE.

٦ – اختر Save As من قائمة File واحفظ هذا البرنامج كملف نصى تحت اسم ملف IN-CLUDE.BAS مع اخلاء الشاشة.

٧ – انتقل إلى الدرس الحادى والعشرين للاستمرار فى تسلسل التعلم.

الدرس الستون

دالة INKEY$

الوصف

تعيد دالة INKEY$ سلسلة من بايت واحد أو اثنين طبقاً للمفتاح المضغوط عليه من لوحة المفاتيح. وتكوين دالة INKEY$ هو كما يلي :

```
INKEY$
```

إذا كان المفتاح المضغوط عليه هو أحد الرموز القابلة للطباعة فيعيد بايت واحد. أما إذا كان المفتاح المضغوط عليه خليطاً من مشاوير مفاتيح مثل Ctrl-X أو Alt-D فنعيد INKEY$ سلسلة من اثنين بايت. وفى هذه الحالة تحتوى البايت الأولى على رمز الفراغ وتحتوى البايت الثانية على رمز الفحص المتسع. يحتوى الملحق B على قائمة برموز فحص لوحة المفاتيح.

ولاتصدر دالة INKEY$ صدى للرموز التى تقرأ من لوحة المفاتيح فى الشاشة. ويمكن أن تستخدم INKEY$ كذلك مع وحدات نمطية أخرى إلا أن لوحة المفاتيح هى الأكثر استخداماً. وتمر كل الرموز التى يتم ادخالها إلى البرنامج باستثناء ما يلى :

Ctrl-Break	ينهى البرنامج
Ctrl-Alt-Del	يعيد بدء عمل النظام
Ctrl-Numlock	يوقف تنفيذ البرامج
PrtSc	يطبع محتويات الشاشة على الطابع

فى البرامج القائمة بذاتها (.EXE) يقرأ تسلسل Ctrl-Break إذا لم يكن خيار d/ محدداً أثناء وقت الترجمة.

التطبيقات

دالة INKEY$ عبارة عن وسيلة مفيدة فى قراءة مفاتيح متسعة من لوحة المفاتيح. وتقدم معالجة سلسلة الرموز التى تعود من الدالة آلية قوية للتحكم فى تداخل البرنامج مع المستفيد.

وفيما يلى بعض الأمثلة لاستخدام INKEY$.

```
DO
LOOP UNTIL (INKEY$ <> "")

Ch$ = INKEY$
Ch1$ = MID$(CH$,1,1): Ch2$ = MID$(CH$,2)
GOSUB ProcessKey
```

أول مثال يستخدم INKEY$ فى الانتظار حتى يتم الضغط على أى مفتاح. ويستخدم المثال الثانى دالة INKEY$ فى الحصول على مشوار مفتاح وتجزئة جزئى السلسلة والتفريع إلى برنامج فرعى سوف يجرى تشغيلاً على جزئى مشوار المفاتيح ويفسر مشوار المفتاح.

عملية تقليدية

هذه العملية توضح كيف يمكن استخدام دالة INKEY$ فى تشغيل مشاوير المفاتيح. يبحث البرنامج بصفة خاصة عن المفاتيح المتسعة التى يتم ادخالها.

ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              INKEY.BAS
'This program demonstrates the use of the INKEY$ function.
CLS
PRINT "This program demonstrates how to find out if a combination"
PRINT "of keystrokes has been entered."
PRINT "The Alt and D through F key combinations are tested."

DO
  DO
    Ch$ = INKEY$
    IF Ch$ <> "" THEN GOSUB InterpretKey
  LOOP UNTIL (Ch$ <> "")

LOOP UNTIL (UCASE$(Ch$) = "Q")
END

InterpretKey:
 Ch1$ = MID$(Ch$, 1, 1): Ch2$ = MID$(Ch$, 2)
 IF ASC(Ch1$) = 0 THEN
    IF ASC(Ch2$) = 32 THEN PRINT "Alt-D was pressed"
    IF ASC(Ch2$) = 18 THEN PRINT "Alt-E was pressed"
    IF ASC(Ch2$) = 33 THEN PRINT "Alt-F was pressed"
 END IF
 RETURN

                              Immediate

Main: INKEY.BAS        Context: Program not running          00001:001
```

٢ – نفذ البرنامج. اضغط على Alt-D و Alt-E اكتب Q للخروج. لاحظ استخدام دالة -IN KEY$ لاكتشاف مشاوير المفاتيح المتسعة. وفيما يلى عينة للتنفيذ :

```
This program demonstrates how to find out if a combination
of keystrokes has been entered.
The Alt and D through F key combinations are tested.
Alt-D was pressed
Alt-E was pressed
Alt-F was pressed





Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الرابع والستين للاستمرار فى تسلسل التعلم.

الدرس الواحد وستون

عبارات INP و OUT و WAIT

الوصف

عبارة INP : تقرأ عبارة INP بايت من بوابة الآلة. وتكوينها هو كما يلى :

```
INP(port)
```

حيث port هو رقم صحيح يحدد بوابة صحيحة لمدخلات ومخرجات الكمبيوتر.

عبارة OUT : ترسل بايت إلى بوابة من بوابات الآلة. وتكوينها هو كما يلى :

```
OUT port,byte
```

حيث port هو بوابة صحيحة لمدخلات ومخرجات الكمبيوتر وجزء byte هو قيمة بيانات ترسل إلى هذه البوابة. وتقع قيمة byte فى المدى من 0 إلى 255.

عبارة WAIT : توقف تنفيذ البرنامج اثناء انتظاره لتطوير نمط بت غير صفرية فى احدى بوابات الآلة. وتكوينها هو كما يلى :

```
t,and-op,xor-op
```

وجزء port عبارة عن رقم صحيح يحدد بوابة صحيحة لمدخلات ومخرجات الكمبيوتر. جزء and-op هو تعبير عددى صحيح مدموج مع البيانات من البوابة فى عملية AND. وجزء xor-op هو تعبير عددى صحيح يكون مدموجاً مع بيانات من البوابة الموجودة فى عملية XOR. تستخدم عمليات بوليان من اليمين إلى اليسار أى إن عملية XOR تؤدى أولا تليها عملية AND. عندما يحذف xor-op فيفترض أنه صفر.

التطبيقات

عبارات INP و OUT و WAIT تستخدم فى التعامل مباشرة مع نظم مكونات الكمبيوتر وعلى هذا فيجب أن تستخدم مع الحرص الشديد. وبسبب الاختلافات فى أجهزة الكمبيوتر الشخصية ونظم مكوناتها، حتى عبر الأجهزة المتوافقة تماماً، فإن المعاملة المباشرة لبوابات مدخلات ومخرجات الكمبيوتر يجب تجنبها كلما كان ذلك ممكناً. وفيما يلى أمثلة لعبارات INP و OUT و WAIT:

```
InByte = INP 32
..
OUT 32,OutByte
..
IF InByte = .. THEN WAIT 32,9
```

عملية تقليدية

تقع مناقشة بوابات المدخلات والمخرجات المختلفة ومعالجتها خارج مدى هذا الكتاب وعلى ذلك فلا يحتوى هذا القسم على مثال لتوضيح هذه الأساسيات.

انتقل إلى الدرس الخامس والتسعين للاستمرار فى تسلسل التعلم.

الدرس الثانى والستون

عبارة INPUT

الوصف

تستخدم عبارة INPUT فى قبول مدخلات من لوحة المفاتيح اثناء تنفيذ البرنامج.

وهذه هى احدى الطرق الأكثر استخداماً فى البيسك فى ادخال قيم المتغيرات. وتكوينها هو كما يلى :

```
INPUT (;) Prompt string (;) (,) variable list
```

الجـزء	الوصـــف
INPUT	كلمة من كلمات بيسك المحجوزة.
(;)	مؤشر اختيارى يتسبب فى أن تظل نقطة البداية موجودة على نفس السطر بعد أن يضغط المستفيد على مفتاح الادخال.
Prompt String	سلسلة اختيارية تعرض كملقن للمدخلات. ويجب أن توضع السلسلة بين علامتى تنصيص مزدوجتين.
(;)	مؤشر اختيارى يتسبب فى طباعة علامة استفهام فى نهاية سلسلة الملقن.
(,)	مؤشر اختيارى يتسبب فى ضغط علامة الاستفهام.
variable list	قائمة متغيرات تقبل فيها المدخلات. ويجب أن تفصل المتغيرات عن بعضها البعض بواسطة فواصل فى عبارة INPUT وفى استجابة INPUT.

توقف عبارة INPUT البرنامج وتطبع علامة استفهام وتنتظر المدخلات من لوحة المفاتيح. ويحدد الضغط على مفتاح الادخال انتهاء المدخلات. فاذا ما كانت القيمة التى يتم ادخالها من نوع غير صحيح أو أنها أكبر من اللازم أو أقل من اللازم فيقدم المترجم رسالة الخطأ التالية :

```
Redo from start
```

فعلى سبيل المثال استجابة الحرف "A" لطلب قيمة عددية ينتج عنه رسالة الخطأ. ويعطى الجدول التالى أمثلة أخرى :

Program line	User response	Valid
INPUT "Enter radius ", R%	12	Yes
	Ac	No
INPUT "Enter List", A$, B$	Hello,Dolly	Yes
	Hello	No
	Dolly	No
INPUT A$,B$,C$,D$	Nuts,in,May,!	Yes
	Nuts in May !	No

يمكن تنقيح البيانات التى يتم ادخالها كاستجابة لعبارة INPUT كما لو كانت مدخلات وقبل الضغط على مفتاح الادخال. وفيما يلى قائمة بالمفاتيح وخليط المفاتيح واجراءاتها.

المفتاح	الوظيفة
السهم الايمن أو Ctrl-\	ينقل نقطة البداية خانة واحدة إلى اليمين.
السهم الأيسر أو Ctrl-J	ينقل نقطة البداية خانة واحدة إلى اليسار.
سهم أيمن -Ctrl أو Ctrl-F	ينقل نقطة البداية كلمة واحدة لليمين.
سهم أيسر -Ctrl أو Ctrl-B	ينقل نقطة البداية كلمة واحدة لليسار.
Home أو Ctrl-K	ينقل نقطة البداية إلى بداية السطر.
End أو Ctrl-N	ينقل نقطة البداية نهاية السطر.
Ins أو Ctrl-R	يغير حالة الادخال من on إلى off والعكس. وعندما تكون حالة الادخال فى وضع on فتضاف الرموز التى يتم الضغط عليها إلى سطر المدخلات وعندما تكون فى وضع off فتمحو الرموز التى يتم الضغط عليها على الحروف السابقة.
Tab أو Ctrl-I	يحرك نقطة البداية بخطوة جدولية إلى اليمين.
Del	لحذف الرمز الذى يقع تحت نقطة البداية.
Backspace أو Ctrl-H	لحذف رمز على يسار نقطة البداية ونقل بقية السطر خانة واحدة إلى اليسار فاذا كانت نقطة البداية فى بداية سطر المدخلات فتحذف الرمز الموجود تحتها.

المفتـــاح	الوظيفـــة
Ctrl-E أو Ctrl-End	لحذف سطر من عند نقطة البداية وحتى نهايته.
Ctrl-U أو Esc	لحذف محتويات سطر بغض النظر عن وضع نقطة البداية.
Ctrl-M أو Enter	لإنهاء المدخلات وتخزينها.
Ctrl- T	لتغيير وضع عرض الوظائف الموجود فى قاعدة الشاشة من وضع on إلى وضع off والعكس.
Ctrl-C أو Ctrl-Break	لانهاء المدخلات والخروج من البرنامج.

التطبيقات

عبارة INPUT هى طريقة سهلة ومتعددة الجوانب لقبول مدخلات من لوحة المفاتيح اثناء تنفيذ البرنامج. وبساطة تكوين العبارة مع قوته يسمح بالابتكارية فى البرمجة. ويمكن أن تقبل المدخلات على هيئة عناصر سلاسل أو عناصر عددية أو منظومات أو سجلات. وفيما يلى قائمة أمثلة.

مثال١

```
INPUT "Enter full name [First.Middle. and Last] ". First$.Middle$.Last$
```

يوجد أكثر من متغير واحد فى قائمة متغيرات هذا المثال. ويجب أن تكون المدخلات عبارة عن ثلاث قيم مفصولة عن بعضها البعض بواسطة فواصل.

مثال٢

```
INPUT "Enter current sales tax ", ST!

CONST MaxRow% = 20
DIM SampleArray(MaxRow%)
FOR Cnt = 1 TO MaxRow%
     INPUT "Next array value ", SampleArray[Cnt]
NEXT Cnt
```

يقبل هذا المثال مدخلات مباشرة فى عناصر المنظومة SampleArray والمعرفة بأنها بها 20 عنصر.

```
TYPE  Client
   Name  AS  STRING*40
   Company AS STRING*40
   Address AS STRING*80
END TYPE
DIM NextClient AS Client
INPUT "Client name and company ", NextClient.Name, NextClient.Company
```

يقبل هذا المثال مدخلات فى نوع سجل يعرفه المستفيد. وتقبل المدخلات داخل كل عنصر بدلاً من قبولها داخل السجل ككل.

وتوضح الأمثلة التالية استخدام الفاصلة المنقوطة والفاصلة فى عبارة INPUT واختلافات أخرى.

```
INPUT ; "Type something ", X$
INPUT " Now type something else ! ", X$
INPUT ; X$
INPUT " Continue ", X$
INPUT "What is your age" Age%
```

عملية تقليدية

استخدام عبارة INPUT موضح فى البرنامج التالى. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                           <Untitled>
CONST MaxFile% = 10
TYPE FileRec
   FileName AS STRING * 15
   FileRef  AS STRING * 40
END TYPE
DIM FileList(MaxFile%) AS FileRec

CLS
PRINT "Filename"; TAB(40); "File Reference": PRINT STRING$(60, "-")
FOR Cnt = 1 TO MaxFile%
   LOCATE Cnt + 2, 1: INPUT FileList(Cnt).FileName
   LOCATE Cnt + 2, 40: INPUT FileList(Cnt).FileRef
NEXT

                            Immediate

Main: <Untitled>      Context: Program not running            00001:012
```

٢ - نفذ البرنامج ولاحظ استخدام عبارة INPUT وكيفية تحكمها فى طريقة ادخال المدخلات فى المنظومة من النوع الذى يعرفه المستفيد والمسماة FileRec. ادخل اسماء ملفات واوصافها كعينة واضغط على Ctrl-Break لإنهاء البرنامج.

```
Filename                                File Reference
-------------------------------------------------------
? Input.Bas                             ? Example INPUT statement
? Record.Inc                            ? Record defns. include file
? Movie.Bas                             ? Moving pictures program
? Sing.Bas                              ? Music program
? Jingle.Bas                            ? Music program
?
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والسبعين للاستمرار فى تسلسل التعلم.

الدرس الثالث والستون

عبارة INPUT#

الوصف

تؤدى عبارة INPUT# نفس الشيء مع البيانات التى يتم ادخالها من ملف والذى تؤديه عبارة INPUT مع البيانات التى يتم ادخالها من لوحة المفاتيح. تقرأ عبارة INPUT# بيانات من ملف تتابعى وتحدد البيانات للمتغيرات. وتكوينها هو كما يلى :

```
INPUT # filenum, variable list
```

جزء filenum هو رقم الملف المحدد بواسطة عبارة OPEN. وجزء variable list هو قائمة بأسماء المتغيرات التى تحدد البيانات المقروءة لها. يجب أن تتفق أنواع أسماء المتغيرات مع البيانات التى تقرأ فيها. ولا تطبع عبارة INPUT# علامة استفهام أثناء التنفيذ. وتكون عناصر البيانات المقروءة مرتبة فى الملف بنفس الطريقة التى تدخل بها كاستجابة لعبارة INPUT#. ولمعرفة تفاصيل أكثر لكيفية ترتيب البيانات ارجع إلى الدرس الثانى والستون. عند قراءة قيم عددية تهمل الفراغات الزائدة ورموز عودة العربة وتغذية السطر وتبدأ القيمة عندما يظهر رمز غير هذه الرموز وتنتهى بظهور فراغ أو فاصلة أو رمز عودة العربة أو رمز تغذية السطر. وعند قراءة قيم سلاسل تهمل الفراغات الزائدة ورموز عودة العربة وتغذية السطر. وعندما تظهر نهاية الملف أثناء قراءة قيمة سلسلة فيعتبر أن عنصر البيانات قد انتهى.

التطبيقات

عبارة INPUT# هى وسيلة مفيدة فى قراءة بيانات من ملف بسرعة وتحديدها لمتغيرات فى نفس الوقت. واستخدام عبارة INPUT# مع ملف بيانات مرتبة منتشر. وفيما يلى بعض الأمثلة.

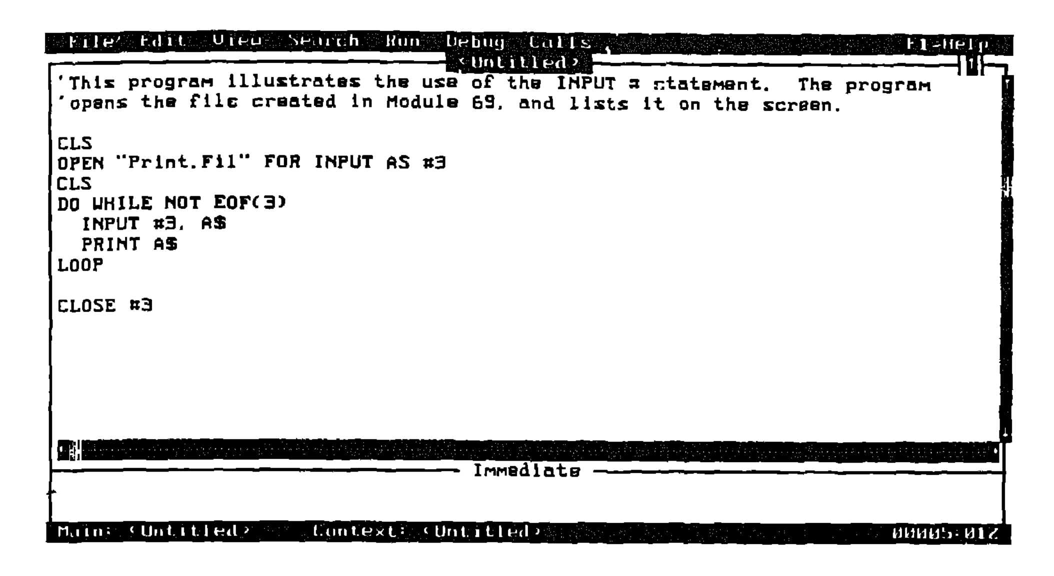
```
OPEN "Score.Dat" FOR INPUT AS #3
...
INPUT #3 Test1, Test2, Test3

OPEN "Ledger" FOR RANDOM AS #2
FIELD #2 12 AS AcctNo, 25 AS AcctName$, 12 AS Amt
...
INPUT #2 AcctNo, AcctName$, Amt
```

عملية تقليدية

توضح هذه العملية عبارة #INPUT. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



٢ – نفذ البرنامج. لاحظ استخدام عبارة #INPUT فى البرنامج. لاحظ بصفة خاصة عبارة IN-PUT# التى تقرأ السطر من الملف وحتى الفاصلة والتى تعتبر كنهاية للبيانات. يتسبب ذلك فى أن تقوم عبارة #INPUT بقراءة البيانات بعد الفاصلة (فى حقل amount) كسجل منفصل.

```
Earth Moving Equipment        12          $120
000
Farm Equipment                22          $85
000
Farm 22    $85
0
SOS







Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج، اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الثامن والستين للاستمرار في تسلسل التعلم.

الدرس الرابع والستون

عبارة INPUT$

الوصف

تقرأ عبارة INPUT$ سلسلة من ملف، وتكوينها هو كما يلى :

```
INPUT$(NumOfChars. #Filenum)
```

يحدد جزء NumOfChars عدد الرموز التى تقرأ من الملف، ويحدد جزء Filenum الملف الذى يقرأ. ويجب أن يكون رقم الملف هو نفس الرقم المستخدم فى عبارة OPEN لهذا الملف. وعندما يكون الملف المحدد مفتوحاً كملف اتصال عشوائى فيجب أن يكون NumOfChars أقل من طول السجل أو يساويه والقيمة التقليدية هى 128 رمزاً. وعندما يحدد أن الاتصال يتم بالملف كملف ثنائى فيجب أن يكون Num Of Chars أقل من أو يساوى 32,767.

فإذا حذف جزء Filenum فتستخدم وحدة المدخلات النمطية فى ادخال البيانات وبدون أن تكون المدخلات قد أعيد توجيهها فتخدم لوحة المفاتيح كوحدة مدخلات نمطية. ارجع إلى دليل DOS لمناقشة اعادة توجيه المدخلات والمخرجات. ولا تصدر عبارة INPUT$ صدى للمدخلات على الشاشة.

التطبيقات

عبارة INPUT$ مفيدة جداً فى قراءة تسلسل من الرموز من ملف أو من وحدة. والمقدرة على قبول مدخلات من وحدة مدخلات نمطية هى ميزة اضافية. وفيما يلى بعض الأمثلة :

مثال١

```
OPEN "Story.Txt" FOR INPUT AS #1
..
PRINT INPUT$(128.#1)
```

مثال٢

```
Line$ = INPUT$(255)
```

يقبل المثال السابق مدخلات من لوحة المفاتيح فى متغير السلسلة Line$.

عملية تقليدية

هذه العملية توضح استخدام عبارة INPUT$. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the use of the INPUT$ statement.
'The program requests a filename and lists that file using the
'INPUT$ statement.
CLS
FILES "????.BAS"
INPUT "Enter file to list: "; FileName$

IF FileName$ <> "" THEN
   OPEN FileName$ FOR INPUT AS #1
   InCh$ = INPUT$(1, #1)
   DO WHILE NOT EOF(1)
      IF (InCh$ <> CHR$(13)) THEN PRINT InCh$;
      InCh$ = INPUT$(1, #1)
   LOOP
   CLOSE #1
ELSE
   PRINT : PRINT "Thank you for participating ..."
END IF
                              Immediate

Main: <Untitled>      Context: Program not running              00018:007
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة INPUT$ فى البرنامج. لاحظ كذلك أن محتويات القرص لا تكون متطابقة مع شكل الشاشة التالى. اختر ملفاً نصياً لعرضه وادخل اسم الملف عند الملقن. اضغط على أى مفتاح للعودة إلى البرنامج.

```
C:\QB
BOX     .BAS      BOX2    .BAS      CASE    .BAS      ASC     .BAS
ABS     .BAS      PLAY    .BAS      OPEN    .BAS
 884736 Bytes free
Enter file to list: ? asc.bas
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
  Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%









Press any key to continue
```

٣ - اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس السابع والسبعين للاستمرار فى تسلسل التعلم.

الدرس الخامس والستون

دالـة INSTR

الوصف

تعيد دالة INSTR موقع أول حدوث لسلسلة داخل سلسلة أخرى. وتكوينها هو كما يلى :

```
INSTR(start,string expression1, string expression2
```

الجـزء	الوصـف
INSTR	كلمة من كلمات بيسك السريع المحجوزة.
start	مؤشر اختيارى يحدد موقع بداية البحث. فإذا لم يتحدد الموقع فيبدأ البحث عند الموقع 1.
string expression 1	السلسلة التى يجرى فيها البحث. ويمكن أن تكون تعبير سلسلة أو متغير أو ثابت.
string expression 2	السلسلة التى يجرى عنها البحث ويمكن أن تكون تعبير سلسلة أو متغير أو ثابت.

التطبيقات

يمكن استخدام عبارة INSTR لفحص داخل محتويات سلسلة بدون استخلاص سلسلة جزئية. والأمثلة التالية توضح بعض طرق استخدامها والنتائج.

مثال١

```
FStr$ = "12345.0909"
PRINT INSTR(FStr$,".")
PRINT INSTR(6,FStr$,".")
```

مثال٢

```
xstr$ = "Peace on earth and good will among men.": spos = 1
CLS
11:
  spos = INSTR(spos + 1, xstr$, " ")
  PRINT spos; " ";
  IF spos > 0 THEN GOTO 11
```

عملية تقليدية

العملية التالية توضح استخدام دالة INSTR. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the INSTR function

DIM A(26) AS INTEGER
Chcnt = 0: CLS
Istr$ = "CAESAR SIPPED HIS SNIFTER AND SEIZED HIS KNEES AND SNEEZED"
IstrZ$ = Istr$
'Count the number of times each character occurs
'Use the ASCII values for the loop count
FOR cZ = 1 TO 26
  Chcnt = 0: SPos = 1   'set the starting position for INSTR

  FOR Cnt = 1 TO LEN(Istr$)

    IF SPos > 0 THEN
       SPos = INSTR(SPos, Istr$, CHR$(cZ + 64))
    END IF

    ' If character found then
    IF SPos > 0 THEN
      A(cZ) = A(cZ) + 1
      'replace the character already found with a non-relevant char.
      MID$(Istr$, SPos) = "x"
      'Display the progress with the string
      LOCATE 1, 1: PRINT Istr$
   END IF

 NEXT Cnt

NEXT cZ

' Print the character count
c = 0
PRINT : PRINT IstrZ$
PRINT : PRINT "The character count for this line is as follows: "
FOR Cnt = 1 TO 26
  'If five counts have been printed, proceed to a new line
  IF c = 5 THEN PRINT : c = 0
  PRINT A(Cnt); "  "; CHR$(64 + Cnt); "'s ";
  c = c + 1
NEXT Cnt

                              Immediate

Main: <Untitled>     Context: Program not running                00054:011
```

٢ - نفذ البرنامج ولاحظ استخدام دالة INSTR فى البرنامج ولاحظ المخرجات.

```
XXXXXX XXXXXX XXX XXXXXXX XXX XXXXXX XXX XXXXX XXX XXXXXXX

CAESAR SIPPED HIS SNIFTER AND SEIZED HIS KNEES AND SNEEZED

The character count for this line is as follows:
 4   A's  0   B's  1   C's  5   D's  10   E's
 1   F's  0   G's  2   H's  5   I's  0    J's
 1   K's  0   L's  0   M's  5   N's  0    O's
 2   P's  0   Q's  2   R's  0   S's  1    T's
 0   U's  0   V's  0   W's  0   X's  0    Y's
 2   Z's

Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. من قائمة File اختر Save واكتب INSERT, BAS كاسم للملف وحدد أن شكل الملف نصى واحفظ هذا البرنامج.

٤ – من قائمة File اختر New.

٥ – انتقل إلى الدرس المائة والرابع عشر للاستمرار فى تسلسل التعلم.

الدرس السادس والستون

دالـة INT

الوصف

تعيد دالة INT أقصى قيمة عددية صحيحة مساوية لمؤشر التعبير العددى أو أقل منه. وتكوينها هو كما يلى :

```
INT(numeric expression)
```

يعود التعبير العددى numeric expression بعد حذف الكسر العشرى منه ومع وضع الاشارة.

التطبيقات

دالة INT هى إحدى دوال التقريب المتاحة فى البيسك السريع وتستخدم فى الحصول على أقرب قيمة عددية صحيحة محددة اشارتها لا تكون أكبر من التعبير العددى. وفيما يلى بعض الأمثلة :

```
PRINT INT(12.5)
```

المخرجات : 12

```
PRINT INT(-7.33)
```

المخرجات : 8-

```
PRINT INT(99.31)
```

المخرجات : 99

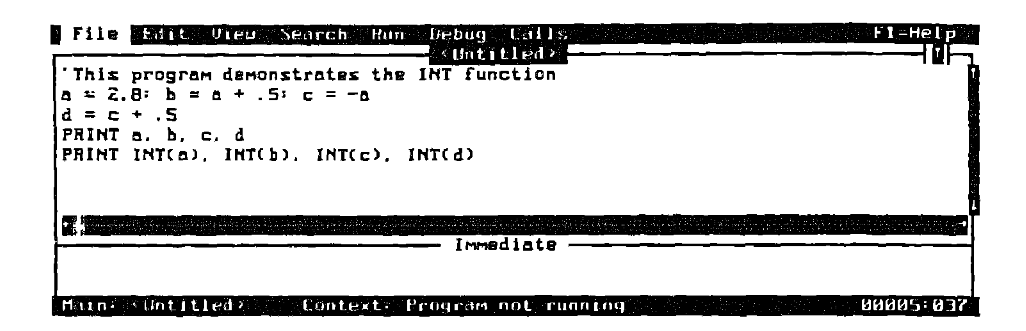
```
PRINT INT(-823.001)
```

المخرجات : 824-

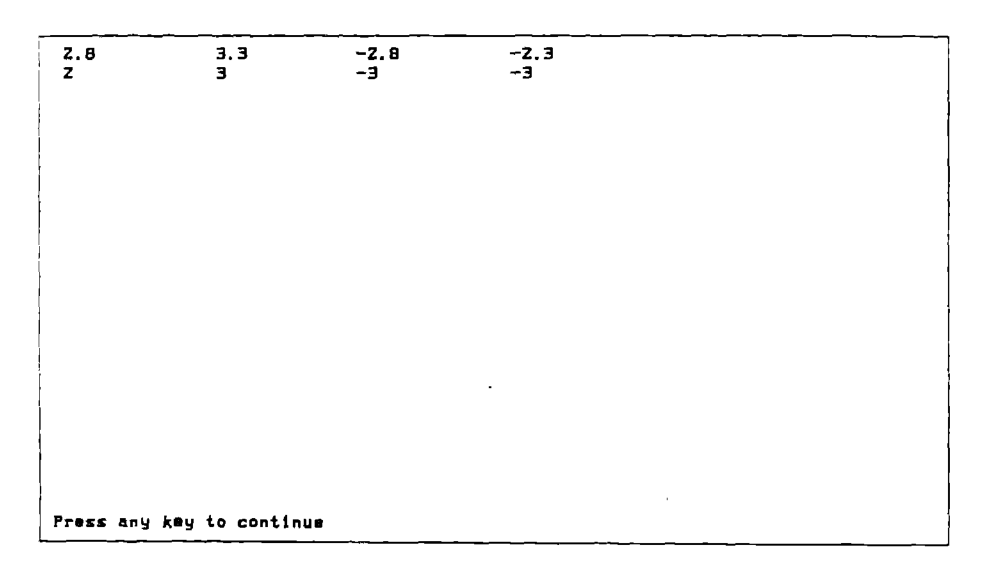
عملية تقليدية

العملية التالية توضح استخدام دالة INT. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج ولاحظ استخدام دالة INT فى البرنامج.



٣ - اضغط على أى مفتاح للعودة إلى البرنامج، من قائمة File اختر New مع عدم حفظ هذا البرنامج.

٤ - انتقل إلى الدرس الخامس للاستمرار فى تسلسل التعلم.

الدرس السابع والستون

عبارة IOCTL ودالة IOCTL$

الوصف

تستخدم عبارة IOCTL ودالة IOCTL$ فى التداخل مع مشغلات الوحدات وتكوينها هو كما يلى:

```
IOCTL #file num. string
IOCTL$ (#file num)
```

ترسل عبارة IOCTL سلسلة بيانات تحكم إلى مشغل الوحدة المحدودة. جزء #filenum هو رقم الملف المستخدم فى عبارة OPEN لفتح الوحدة. وجزء string هو بيانات التحكم التى ترسل إلى مشغل الوحدة كأمر. ويمكن أن يصل طول السلسلة حتى 32,767 رمزاً كحد أقصى.

وتستقبل دالة IOCTL$ سلسلة بيانات تحكم من مشغل وحدة معين. وجزء #filenum هو نفسه مثل ما هو موجود فى عبارة IOCTL.

وفيما يلى المتطلبات اللازمة لعمل IOCTL و IOCTL$.

- يجب أن يكون مشغل الوحدة معداً.
- يجب أن يكون مشغل الوحدة قادراً على تشغيل سلاسل IOCTL، ويمكن الحصول على هذه المعلومات من الوثائق الخاصة بمشغل الوحدة. كما يمكن الحصول كذلك على معلومات عن دعم IOCTL من خلال ازعاجات DOS (DOS interrupts). وللمزيد من المعلومات عن استدعاءات دوال DOS وازعاجاته ارجع إلى الدليل التقنى لنظام DOS.
- يجب أن يؤدى بيسك السريع عبارة OPEN ويجب أن يظل الملف مفتوحاً.

وحدات بيسك السريع : LPT1 و COM1 و COM2 و :SCRN و :CONS ووحدات مجموعة DOS من : A حتى :Z لا تدعم IOCTL.

التطبيقات

تستخدم عبارة IOCTL ودالة IOCTL$ فى تطبيقات برمجة مشغلات الوحدات ومثل هذه البرمجة تكون محددة بوحدة معينة. ومناقشة مشغلات الوحدات تقع خارج مدى هذا الكتاب.

انتقل إلى الدرس الأربعين للاستمرار فى تسلسل التعلم.

الدرس الثامن والستون

عبارات KEY

الوصف

يتعامل هذا الجزء مع عبارات المفاتيح KEY المستخدمة فى تحديد قيم لمفاتيح الوظائف وانتاج تسلسلات مفاتيح يعرفها المستفيد وسرد تحديدات المفاتيح وتمكين وإلغاء إمكانية وايقاف تصيد أحداث KEY. والعبارات لها التكوين التالى :

```
KEY n, string
KEY LIST
KEY ON
KEY OFF
KEY (n) ON
KEY (n) OFF
KEY (n) STOP
```

عبارة KEY n, string : تحدد هذه العبارة تعبير سلسلة لمفتاح وظيفة معين. وجزء n هو عدد يقع بين 1 و 10 أو 30 أو 31. تعنى الاعداد من 1 إلى 10 مفاتيح الوظائف من F1 حتى F10. كما أن 30 و 31 يحددان مفتاحى الوظائف F11 و F12 فى لوحة مفاتيح AT الموسعة. جزء string هو متغير سلسلة أو ثابت يحدد إلى مفتاح معين. يتسبب ذلك فى انتاج السلسلة عندما يكون المفتاح مضغوطاً عليه. ويمكن تحديد قيمة السلسلة بحد أقصى 15 رمزاً فإذا كان طول السلسلة أكبر من ذلك فتهمل الرموز الزائدة عن 15 رمزاً. وتحديد سلسلة فارغة لمفتاح معين يجعل المفتاح غير قادر على أداء التحديد السابق له. وتسمى هذه التحديدات بالمفاتيح الناعمة soft keys.

عبارة KEY LIST : تتسبب هذه العبارة فى طباعة تحديدات السلاسل لمفاتيح الوظائف على الشاشة. وتعرض كل الرموز (15 رمزاً) الموجودة فى السلسلة والمحددة لكل مفتاح على الشاشة.

عبارة KEY ON : تتسبب هذه العبارة فى عرض تحديدات المفاتيح الناعمة فى قاعدة الشاشة. ويعرض أول ستة رموز من السلسلة فقط.

عبارة KEY OFF : تتسبب هذه العبارة فى حذف عرض KEY ON من على الشاشة لجعل هذا السطر من الشاشة متاحاً للبرنامج.

عبارات اصطياد نشاط المفتاح EVENT TRAPPING KEY STATEMENTS:

اصطياد نشاط المفتاح متاح لمفاتيح الوظائف من 1 إلى 12 ولمفاتيح التحكم فى نقطة البداية والمفاتيح التى يعرفها المستفيد وهى 10. قيمة الجزء n هى كما يلى :

n	المفاتيح
1-10	مفاتيح الوظائف من 1 إلى 10.
11	حركة نقطة البداية لأعلى.
12	حركة نقطة البداية لليسار.
13	حركة نقطة البداية لليمين.
14	حركة نقطة البداية لأسفل.
15-25	المفاتيح التى يعرفها المستفيد.
30,31	مفتاحى F11 و F12.

وفيما يلى التكوين المستخدم لتعريف المفاتيح التى يحددها المستفيد :

```
KEY n, CHR$(kbd flag) + CHR$(scan code)
```

حيث n هو رقم يتراوح من 15 إلى 25 وجزء kbd flag هو رمز ينتج عندما يتم الضغط على مفتاح متسع وجزء scan code هو رمز فحص فى لوحة المفاتيح لمفتاح محدد. ويسمح هذا بتعريف اكتشاف واصطياد الخليط من مشاوير المفاتيح. وفيما يلى قائمة بقيم إشارة kbd التى يمكن استخدامها فى الخليط.

kbd flag	المفتاح
&h00	لاتوجد اشارة لوحة مفاتيح
&h01-&h03	مفاتيح ترحيل يسار أو يمين
&h04	مفتاح تحكم
&h08	مفتاح تبديل
&h20	مفتاح اغلاق الأعداد
&h40	مفتاح اغلاق الحروف الصغيرة
&h80	مفتاح لوحة مفاتيح AT المتسعة

وفيما يلى قائمة برموز فحص لوحة المفاتيح، لاحظ أن الرموز موجودة بالنظام السادس عشر.

Key	*Code*	*Key*	*Code*	*Key*	*Code*
Esc	01	Ctrl	1D	Spacebar	39
!,1	02	A	1E	Capslock	3A
@,2	03	S	1F	F1	3B
#,3	04	D	20	F2	3C
$,4	05	F	21	F3	3D
%,5	06	G	22	F4	3E
^,6	07	H	23	F5	3F
&,7	08	J	24	F6	40
*,8	09	K	25	F7	41
(,9	0A	L	26	F8	42
),0	0B	:,;	27	F9	43
_,-	0C	",'	28	F10	44
+,=	0D	~,'	29	Numlock	45
Left	0E	Left Shft	2A	ScrlLock	46
Tab	0F	¦,\	2B	Home,7	47
Q	10	Z	2C	Up,8	48
W	11	X	2D	PgUp,9	49
E	12	C	2E	−	4A
R	13	V	2F	Left,4	4B
T	14	B	30	5	4C
Y	15	N	31	Right,6	4D
U	16	M	32	+	4E
I	17	< ,,	33	End,1	4F
O	18	>,.	34	Down,2	50
P	19	?,/	35	PgDn,3	51
{,[	1A	Right Shf	36	Ins,0	52
},]	1B	PrtScr,*	37	Del,.	53
Return	1C	Alt	38		

عبارة KEY (n) ON: تمكن هذه العبارة اصطياد النشاط للمفتاح n. وعندما يكون هذا المفتاح مضغوطاً عليه فيميزه البرنامج ويمكن أن يتأثر به.

عبارة KEY (n) OFF : تلغى هذه العبارة من مقدرة اصطياد النشاط للمفتاح n. وتتسبب في ألا يميز البرنامج متى يكون هذا المفتاح مضغوطا أثناء تنفيذ البرنامج. ويمكن على أية حال اكتشاف المفتاح باستخدام طرق أخرى.

عبارة KEY (n) STOP: توقف هذه العبارة اصطياد نشاط المفتاح n. والأنشطة التى تنفذ بعد هذه العبارة لا يحدث لها اصطياد لكن يمكن تذكرها. بعد تنفيذ عبارات KEY (n) ON متتالية يتم تشغيل النشاط.

التطبيقات

تقدم عبارات KEY طريقة قوية وشاملة للتعامل مع مدخلات لوحة المفاتيح. وفى أى تطبيق متداخل مطور يوصى بشدة باستخدام هذه العبارات. وفيما يلى أمثلة لعبارات KEY :

```
KEY ON
KEY 1, "Help"
KEY 2, "Main Menu"
KEY 3, "Create"
KEY 4, "Edit"
KEY 5, "Browse"
CLS
KEY LIST
INPUT t$
```

وعندما تنفذ هذه العبارات ينتج عن تنفيذها شاشة تشبه ما يلى :

```
F1   Help
F2   Main Menu
F3   Create
F4   Edit
F5   Browse
F6
F7
F8
F9
F10
?









1Help  2Main M  3Create  4Edit  5Browse  6        7        8        9        10
```

وقد كانت هناك عبارة INPUT بحيث يمكن ملاحظة الشاشة الناتجة. وفيما يلى أمثلة لعبارات KEY.

```
KEY 20, Chr$(&h0) + Chr$(&h01)  'define the Esc key
KEY 15, Chr$(&h05) + Chr$(&h01) 'define Ctrl-Shift-Esc
KEY (20) ON: KEY (15) ON        'enable event trapping for both

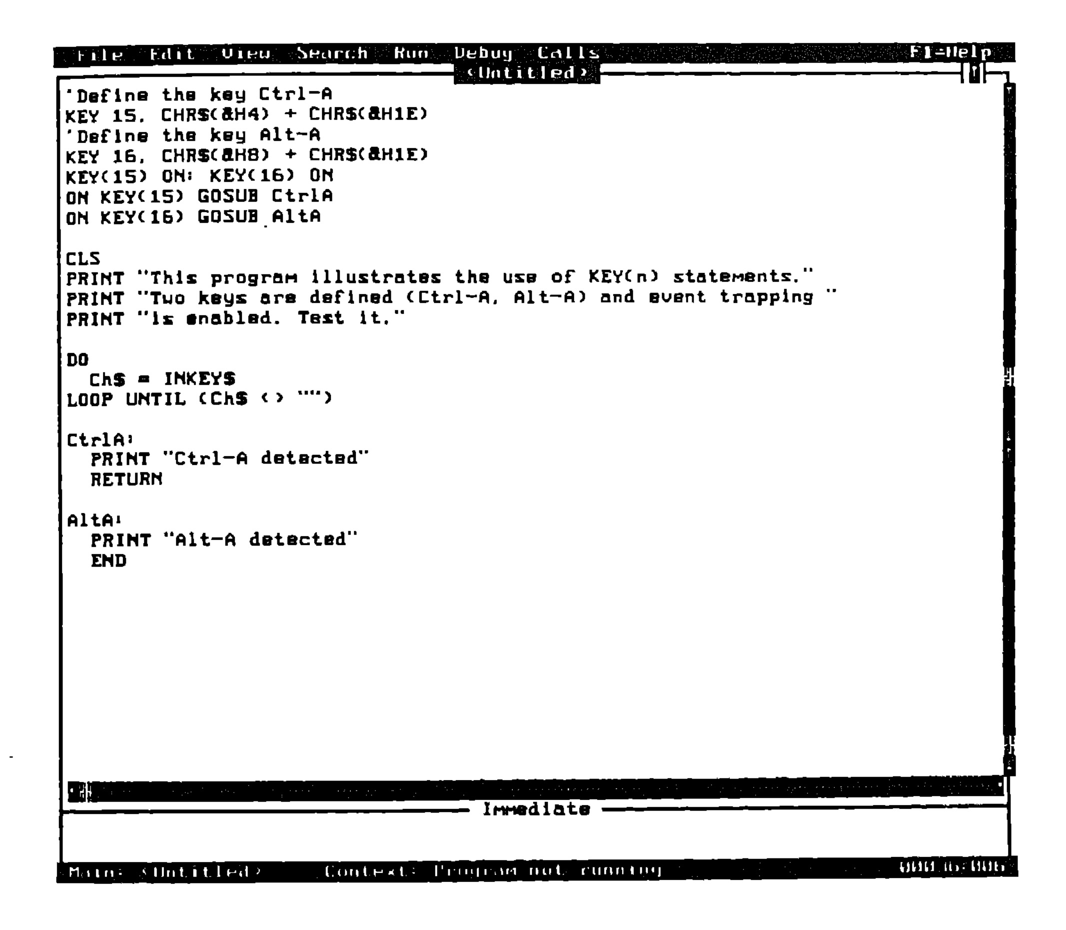
      ON KEY (20) GOSUB Abort
      ON KEY (15) GOSUB ChangeModes:
           ..
           KEY (20) OFF        'disable the Esc key
           RETURN
```

عملية تقليدية

هذه العملية عبارة عن توضيح موجز لعبارات KEY (n). يعرف البرنامج مشوارى مفتاحين Ctrl-A و Alt-A ويمكن من اصطياد النشاط لكل من المفتاحين ويوضح أنهما يعملان.

ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



```
'Define the key Ctrl-A
KEY 15, CHR$(&H4) + CHR$(&H1E)
'Define the key Alt-A
KEY 16, CHR$(&H8) + CHR$(&H1E)
KEY(15) ON: KEY(16) ON
ON KEY(15) GOSUB CtrlA
ON KEY(16) GOSUB AltA

CLS
PRINT "This program illustrates the use of KEY(n) statements."
PRINT "Two keys are defined (Ctrl-A, Alt-A) and event trapping "
PRINT "is enabled. Test it."

DO
  Ch$ = INKEY$
LOOP UNTIL (Ch$ <> "")

CtrlA:
  PRINT "Ctrl-A detected"
  RETURN

AltA:
  PRINT "Alt-A detected"
  END
```


٢ – نفذ البرنامج، اضغط على Ctrl-A و Alt-A لفصل البرنامج. لاحظ استخدام عبارات KEY (n) فى البرنامج، وفيما يلى عينة للتنفيذ :

```
This program illustrates the use of KEY(n) statements.
Two keys are defined (Ctrl-A, Alt-A) and event trapping
is enabled. Test it.
Ctrl-A detected
Alt-A detected




Press any key to continue
```

٣ - ارجع إلى البرنامج واخل الشاشة دون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس العاشر للاستمرار فى تسلسل التعلم.

الدرس التاسع والستون

عبارة KILL

الوصف

تحذف عبارة KILL ملفاً محدداً من القرص، وتكوينها هو كما يلى :

```
KILL file specification
```

يقدم جزء file specification معلومات عن موقع الملف المطلوب ويشمل بصورة اختيارية المسار. ويجب أن تكون مواصفات الملف موضوعة بين علامتى تنصيص. ويمكن اتباع تكوين مسار DOS. ويمكن استخدام اسم الملف الموجود فى مواصفات الملف مع الرموز wild-card من DOS فى حذف أكثر من ملف واحد فى نفس الوقت. فإذا ما حاولت أن تحذف "KILL" ملفاً مفتوحاً كمدخلات أو كمخرجات فينتج بيسك السريع رسالة خطأ تفيد بأن الملف مفتوح بالفعل.

التطبيقات

تستخدم عبارة KILL فى حذف ملفات أثناء تنفيذ البرنامج، وهى مفيدة بصفة خاصة عندما ينتج البرنامج ملفات مؤقتة أثناء التنفيذ. وفيما يلى بعض الأمثلة :

```
KILL "SCORE.DAT"
KILL "C:\MASM\CLRSCR.BAK"
KILL "C:\C\*.BAK"
KILL "SCORE?.*"
KILL "SCORE*.*"
```

لاحظ الفرق فى الرموز الخاصة بين أخر مثال والمثال السابق له. مواصفة الملف "SCORE?.*" تتسبب فى حذف كل الملفات التى تبدأ بكلمة SCORE يليها أى رمز آخر أو أى عدة رموز أخرى ولها أى اتساع.

عملية تقليدية

توضح العملية التالية استخدام عبارة KILL. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
CLS
FILES
PRINT "Enter the filename to delete. Press ENTER to end."
INPUT "File name: "; F$
IF F$ <> "" THEN KILL F$

                              Immediate

Main: <Untitled>      Context: Program not running              00006:001
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة KILL فى البرنامج. اضغط على مفتاح الادخال لإنهاء البرنامج دون أن تحذف أى ملف.

```
C:\QB
         .     <DIR>         ..     <DIR> BC       .EXE      QB       .EXE
SETUP    .BAT     SETUP1   .BAT      PACKING  .LST      README   .DOC
BRUN40   .EXE     BRUN40   .LIB      BCOM40   .LIB      BQLB40   .LIB
SAMPLE   .BAS     LIB      .EXE      LINK     .EXE      QB       .HLP
MOUSE    .COM     NOEM     .OBJ      REMLINE  .BAS      SORTDEMO.BAS
TORUS    .BAS     ABSOLUTE.ASM       INTRPT   .ASM      QB       .LIB
QB       .QLB     QB       .PIF      QB       .BI       NOCOM    .OBJ
DEMO1    .BAS     DEMO2    .BAS      DEMO3    .BAS      QBHERC   .COM
FIXSHIFT.COM      INCHZCM  .BAS      BOX      .BAS      BOX      .OBJ
BOX      .EXE     EXE      .MAP      BOX2     .BAS      PRINT    .BAS
STRING   .BAS     IFTHEN   .BAS      CASE     .BAS      ASC      .BAS
ULCASE   .BAS     LRTRIM   .BAS      ABS      .BAS      RANDOM   .BAS

 897024 Bytes free
Enter the filename to delete. Press ENTER to end.
File name: ?

Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس المائة والستين للاستمرار فى تسلسل التعلم.

الدرس السبعون

الأسماء LABELS

الوصف

الأسماء Labels هى أسماء تستخدم كمحددات لمواقع. وتستخدم فى تعريف موقع أو سطر فى البرنامج للاتصال به فيما بعد. وتستخدم أساساً بغرض عمل تفريعات. وأما أن تكون الأسماء فى بيسك السريع عددية أو حرفية عددية. ولايمكن للأسماء أن تحتوى على أى رموز غير الأعداد والحروف الهجائية أى إنها لا يمكن أن تحتوى على أى رموز خاصة.

يمكن أن يبدأ الاسم برقم أو بحرف ولا يمكن أن يزيد طوله عن 40 خانة وينتهى دائماً بنقطتين رأسيتين. ويمكن ألا تتبع النقطتان الرأسيتان الاسم مباشرة. ويسمح بوجود اسم label واحد فقط على سطر واقعى واحد. وفيما يلى بعض الأمثلة :

```
ShowPrompt:
Menu01:
099:
FileKill:
```

التطبيقات

تستخدم الأسماء labels كأدلة لعبارات GOTO و GOSUB. ولا يمكن استخدامها مع عبارات IF..THEN. دليل تعريف السطر فى عبارة IF..THEN يجب أن يكون رقم سطر. استخدام اسماء labels ذات معنى فى برنامج البيسك يجعل الشفرة أسهل فى قراءتها عن استخدام أرقام الأسطر.

عملية تقليدية

توضح هذه العملية أحد الاستخدامات الممكنة للأسماء labels. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
CLS
Again:

INPUT "Enter selection (1..3)", S

IF S = 1 THEN GOTO Choice1
IF S = 2 THEN GOTO Choice2
IF S = 3 THEN GOTO Choice3
GOTO Again

Choice1:
PRINT "Hello from label Choice1": END

Choice2:
PRINT "Hello from label Choice2": END

Choice3:
PRINT "Hello from label Choice3": END

                              Immediate

Main: <Untitled>      Context: Program not running                 00001:004
```

٢ – اضـغط على Shift-F5 لتنفيذ البرنامـج. اكتب 2 واضـغط على مفتاح الادخال للحصـول على المخرجات التالية :

```
Enter selection (1..3) 2
Hello from label Choice2




Press any key to continue
```

٣ – اضـغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ – احـفظ البـرنامج فى شكل نصـى تحت اسم LABEL.BAS. اضـغط على Alt-F واكتب S لحـفظ صـندوق الحـوار. اكتب LABEL.BAS كـاسم للملف واضـغط على Tab واستخدم مفاتيح الأسهم لتحديد أن الشكل نصى. اضغط على Tab مرة أخرى ثم اضغط على قضيب المسافات.

٥ – انتقل إلى الدرس التاسـع والسبعين للاستمرار فى تسلسل التعلم.

الدرس الحادى والسبعون

دالتا LBOUND و UBOUND

الوصف

تعيد دالة LBOUND أقل دليل لبعد معين لمنظومة. وتعيد دالة UBOUND أكبر دليل لبعد معين لمنظومة. والتكوين هو كما يلى :

```
LBOUND(array,dimension)
UBOUND(array,dimension)
```

جزء array هو متغير منظومة وجزء dimension هو البعد المراد تحديد أكبر أو أصغر دليل له. ويترك جزء dimension عندما يكون للمنظومة بعد واحد فقط. وتعيد دالة LBOUND القيمة 1 أو القيمة 0 بصورة تقليدية طبقاً لأعداد أساس الخيار OPTION BASE.

التطبيقات

تستخدم دوال LBOUND و UBOUND فى ايجاد الحدود الدنيا والعليا لبعد المنظومة. وهذا مفيد فى تقويم حجم المنظومة وعندما لا تكون المنظومة موضحة محلياً وعندما لا يكون للبرنامج الفرعى اتصال بالتوضيحات. وفيما يلى بعض الأمثلة :

مثال١

```
DIM Rt(100,2)
ArrLow = LBOUND(Rt,1)
ArrHigh = UBOUND(Rt,1)
```

مثال٢

```
DIM Qt(12,10,99)
AL = LBOUND(Qt,2)
AL3 = LBOUND(Qt,3)
AH1 = UBOUND(Qt1)
```

مثال٣

```
OPTION BASE 1
DIM Sngl(26)
SnglLow = LBOUND(Sngl)
SnglHigh = UBOUND(Sngl)
```

عملية تقليدية

هذه العملية توضح استخدام دالتى LBOUND و UBOUND. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'The following program demonstrates the use of the UBOUND and LBOUND
'statements. The program loads two sets of array values and
'finds the minimum and maximum values in those arrays.

DECLARE SUB FindMinMax (A%(), MinVal, MaxVal)

Max = 15
DIM A%(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A%(Cnt)
  NEXT
  RETURN


Start:
  CLS
  GOSUB LoadArray
  PRINT "First pass"
  CALL FindMinMax(A%(), MinVal, MaxVal)
  READ Max
  REDIM A%(Max)
  GOSUB LoadArray
  PRINT "Second pass"
  CALL FindMinMax(A%(), MinVal, MaxVal)

DATA 12,23,33,43,1,56,98,656,323,44,3,88,67,54,10
DATA 10
DATA 8,89,76,54,23,32,12,4,33,54
```

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                    <Untitled>:FindMinMax
SUB FindMinMax (A%(), MinVal, MaxVal)
 MinVal = A%(1): MaxVal = A%(1)
 FOR Cnt = LBOUND(A%) + 1 TO UBOUND(A%)
   IF MinVal > A%(Cnt) THEN
      MinVal = A%(Cnt)
   END IF
   IF MaxVal < A%(Cnt) THEN
      MaxVal = A%(Cnt)
   END IF
 NEXT
 PRINT "Minimum value in array: "; MinVal, "Maximum value in array: "; MaxVal
END SUB

                           Immediate

Main: <Untitled>     Context: Program not running          00011:078
```

٢ – نفذ البرنامج. لاحظ استخدام دالتى LBOUND و UBOUND فى البرنامج.

```
First pass
Minimum value in array:  1  Maximum value in array:  656
Second pass
Minimum value in array:  4  Maximum value in array:  89




Press any key to continue
```

٣ – ارجع إلى البرنامج. احفظ البرنامج على أنه برنامج نصي وله الاسم LBOUND.BAS مع اخلاء الشاشة.

٤ – انتقل إلى الدرس المائة والسابع والثلاثين للاستمرار في تسلسل التعلم.

الدرس الثانى والسبعون

دالتا LCASE$ و UCASE$

الوصف

تحول دالتا LCASE$ و UCASE$ قيم سلاسل إلى الحالة السفلية (حروف صغيرة) والحالة العلوية (حروف كبيرة) على التوالى. وتكوين الدالتين هو كما يلى :

```
LCASE$(String exp)
UCASE$(String exp)
```

أجزاء string exp فى كل من التكوينين هى تعبيرات سلاسل. تقبل الدالتان كل من السلاسل ثابتة الطول ومتغيرة الطول كمؤشرات لها. ويمكن أن يكون تعبير السلسلة ثابت سلسلة أو متغير سلسلة أو أى تعبير ينتج عنه سلسلة.

التطبيقات

دوال LCASE$ و UCASE$ تكون أكثر نفعاً فى مقارنات سلاسل من المهام فيها تمييز حالة الحروف (سفلية أو علوية). وفيما يلى بعض الأمثلة لاستخداماتهما :

مثال١

```
DO WHILE UCASE$(Choice$) <> "Q"
    ..
LOOP
```

مثال٢

```
IF UCASE$(Response$) = UCASE$(Option$) THEN
    ..
END IF
```

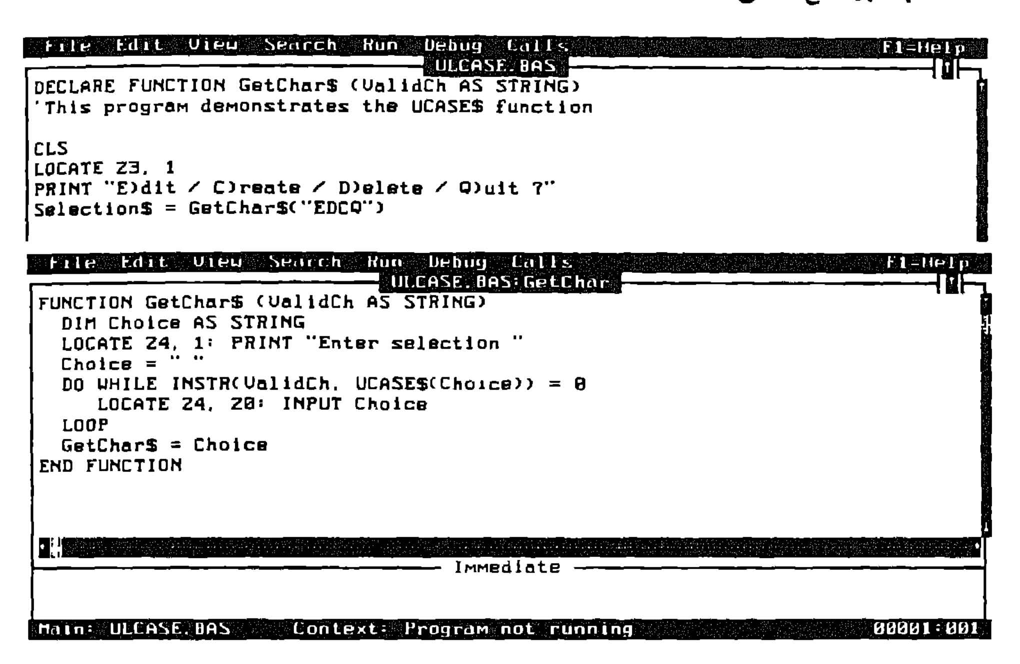
مثال٣

```
SELECT CASE LCASE$(Entry$)
     CASE "1"
          ..
END SELECT
```

عملية تقليدية

تعطى هذه العملية برنامجاً لتوضيح دالتى LCASE$ و UCASE$ لمقارنات سلاسل تهتم بالحالة. وتقبل الدالة المعرفة فى هذا البرنامج مدخلات وتقارنها مع الاختيارات المسموح بها وتعيد القيمة التى أدخلت إلى العبارة المنادية. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         ULCASE.BAS
DECLARE FUNCTION GetChar$ (ValidCh AS STRING)
'This program demonstrates the UCASE$ function

CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"
Selection$ = GetChar$("EDCQ")

File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                      ULCASE.BAS:GetChar
FUNCTION GetChar$ (ValidCh AS STRING)
  DIM Choice AS STRING
  LOCATE 24, 1: PRINT "Enter selection "
  Choice = " "
  DO WHILE INSTR(ValidCh, UCASE$(Choice)) = 0
     LOCATE 24, 20: INPUT Choice
  LOOP
  GetChar$ = Choice
END FUNCTION

                          Immediate

Main: ULCASE.BAS      Context: Program not running            00001:001
```

٢ - نفذ البرنامج ولاحظ سلوك البرنامج مع قبوله للمدخلات. ادخل حرفاً واحداً من الاختيار لإيقاف البرنامج.

```
E)dit / C)reate / D)elete / Q)uit ?
Enter selection
                  ? f
                  ? s
                  ? q

Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج.

٤ - من قائمة File اختر Save واكتب UCASE.BAS كاسم للملف. حدد أن شكل الملف هو نصى واحفظ الملف.

٥ - انتقل إلى الدرس السادس والثمانين للاستمرار فى تسلسل التعلم.

الدرس الثالث والسبعون

دالــة LEFT$

الوصف

تعيد دالة LEFT$ عدد الرموز المحدد فى أقصى اليسار من مؤشر تعبير السلسلة. وتكوينها هو كما يلى :

```
LEFT$(String expression, num)
```

الوصــف	الجــزء
كلمة من كلمات بيسك السريع المحجوزة.	LEFT$
السلسلة التى يعود منها العدد المحدد للرمز الموجود فى أقصى اليسار. ويمكن أن يكون تعبير سلسلة أو متغير أو ثابت.	string expression
عدد الرمـوز المطلوب اعادتها. فإذا كان num أكبر من طول السلسلة فتعود كل السلسلة.	num

التطبيقات

دالة LEFT$ هى وسيلة أخرى لتشغيل سلاسل فى بيسك السريع. وتستخدم فى الحصول على سلسلة جزئية من تعبير سلسلة طبقاً لاتجاه محدد. ودائماً ما تكون السلسلة الجزئية التى تعود هى عدد الرموز المحدد فى أقصى اليسار. وفيما يلى بعض الأمثلة :

مثال١

```
Name$ = "Marmaduke Blenkinsop"
PRINT LEFT$(Name$,INSTR(Name$," "))
```

يبحث هذا المثال عن أول حدوث لفراغ ويطبع كل الرموز التى تسبق هذا الفراغ الأول.

مثال٢

```
Phrase$ = "Stitch in time"
PRINT LEFT$(Phrase$,40)
```

يطبع هذا المثال محتويات السلسلة Phrase$ نظراً لأن عدد الرموز المحدد فى قائمة LEFT$ أكثر من LEN (Phrase$).

مثال٣

```
Phrase$ = "Saves nine"
PRINT LEFT$(Phrase$,0)
```

تكون مخرجات هذا المثال عبارة عن سلسلة فارغة لأن عدد الرموز المحددة هو صفر.

عملية تقليدية

لتجربة برنامج بسيط يوضح دالة LEFT$ استمر على النحو التالى. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                             <Untitled>
Var$ = "ABRACADABRA"
PRINT : PRINT Var$
FOR Cnt = 1 TO LEN(Var$)
  PRINT LEFT$(Var$, Cnt)
NEXT Cnt

                             Immediate

Main: <Untitled>      Context: Program not running            00001:001
```

٢ – نفذ البرنامج ولاحظ المخرجات واستخدام LEFT$ فى البرنامج.

```
ABRACADABRA
A
AB
ABR
ABRA
ABRAC
ABRACA
ABRACAD
ABRACADA
ABRACADAB
ABRACADABR
ABRACADABRA







Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الثانى والسبعين للاستمرار فى تسلسل التعلم.

الدرس الرابع والسبعون

دالة LEN

الوصف

تعطى دالة LEN طول السلسلة المحددة أو عدد البايت اللازم للمتغير المحدد. وتكوينها هو كمايلى :

```
LEN(String expression)   or   LEN(Variable)
```

الجـزء	الوصـف
LEN	كلمة من كلمات بيسك المحجوزة.
String expression	تعبير سلسلة أو ثابت حرفى.
variable	اسم متغير بيسك صحيح ويمكن أن يكون من أى نوع

التطبيقات

يختلف استخدام دالة LEN. ففى المثال المقدم فى الدرس الثالث تستخدم LEN فى تضبيط النص فى منتصف الصندوق. وتشمل الاستخدامات الأخرى حدود عدادات الدورات وعمل تشكيلات المخرجات وعمل التشكيلات النصية. ويمكن لدالة LEN، عند استخدامها فى الحصول على بايت الذاكرة اللازم لأحد المتغيرات، ان تستخدم فى تقويم متطلبات تقويم ذاكرة البرنامج وحجم السجل للاتصال بملف عشوائى. وفيما يلى بعض الأمثلة :

مثال١

```
PRINT "Long word is " LEN("Supercalifragilistic");
      "chars. long"
```

مثال٢

```
LStr$ = "New kid on the block. Eh?"
FOR Cnt = 1 TO LEN(LStr$)
  . .
NEXT Cnt
```

يستخدم هذا المثال القيمة التى تعود من LEN كقيمة تحكم فى دورة FOR.

مثال٣

```
LStr$ = "Move it to the right!"
PRINT TAB(79-LEN(LStr$)) LStr$
```

مثال٤

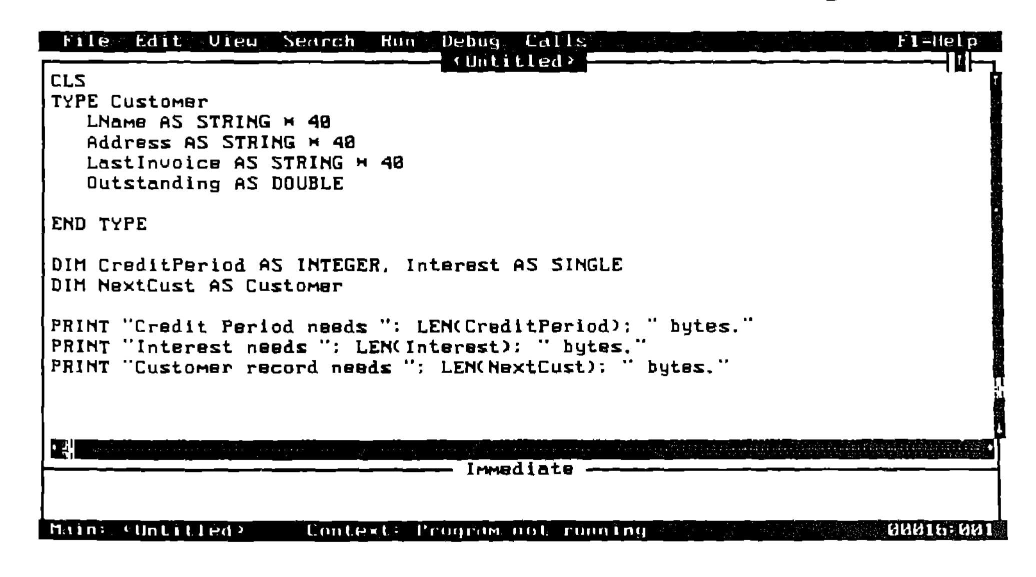
يعكس هذا المثال سلسلة مدخلات ويوضح أحد استخدامات دالة LEN.

```
LStr$ = "ABLE WAS I ERE I SAW ELBA"
PRINT LStr$
FOR SPos = LEN(LStr$) TO 1 STEP -1
   PRINT MID$(LStr$,SPos,1);
NEXT SPos
```

عملية تقليدية

يوضح البرنامج التالى كيفية ايجاد البايت اللازمة بواسطة متغير. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ استخدام دالة LEN فى الحصول على حجم البايت لمتغيرات يعرفها المستفيد.

```
Credit Period needs  2  bytes.
Interest needs  4  bytes.
Customer record needs  128  bytes.





Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس السابع والثمانين للاستمرار فى تسلسل التعلم.

الدرس الخامس والسبعون

عبارة LET

الوصف

تستخدم عبارة LET فى تحديد قيمة لمتغير، ويمكن أن تكون القيمة تعبيراً أو ثابتاً ويجب أن تكون من نفس نوع المتغير. وتكوينها هو كما يلى :

```
LET variable=expression
```

كلمة بيسك السريع LET المحجوزة تكون اختيارية فى التكوين وتكفى علامة التساوى. تتحدد قيمة التعبير الموجود فى الطرف الأيمن لاسم المتغير الموجود فى الطرف الأيسر. والمتغير هو أى متغير بيسك سريع صحيح ويمكن للتعبير أن يكون عددياً أو سلسلة أو حرفياً. فى حالة متغيرات من النوع الأساسى أو الأولى فإذا كان المتغير يشار إليه لأول مرة فى البرنامج فيلزم للاشارة أن توضح المتغير. ويصبح المتغير نشطاً فى أول مرة تستخدمه. ويعمل التحديد بطريقة طيبة حتى إذا لم يستخدم الفعل LET. والتحديد A=10 صحيح من ناحية التكوين ولاينتج البرنامج خطأ أثناء ترجمته وينفذ دون أن يتأثر بعدم وجود الفعل LET.

التطبيقات

استخدم LET فى تعديل قيمة المتغير الحالية أو فى وضع قيمة ابتدائية معينة للمتغير. وفيما يلى بعض الأمثلة :

```
LET A = 23: LET B = 15
LET Question$ = "What's up, Doc?"
NewQuestion$ = "What's up..DUCK?"
```

عملية تقليدية

توضح وتحدد قيماً لمتغيرات فى هذا القسم سواء كان ذلك مع ظهور أو مع عدم ظهور الفعل LET. ابدأ بتحميل بيسك السريع. (ارجع إلى البدء فى الدرس الثالث وفى الملحق B للمزيد من المعلومات عن بدء بيسك السريع).

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                          <Untitled>
LET LastName$ = "Blenkinsop"
FirstName$ = "Cecil"
PRINT Q, LastName$ "," FirstName$

                          Immediate

Main: <Untitled>        Context: Program not running               00001:014
```

٢ – اضغط على Shift-F5 لتنفيذ البرنامج. تظهر المخرجات على النحو التالى :

```
 0             Blenkinsop,Cecil

 Press any key to continue
```

لاحظ المخرجات. تم التحديد للمتغير $FirstName بدون عبارة LET. لا يستخدم المتغير Q فى أى مكان باستثناء استخدامه فى عبارة PRINT ويكون له قيمة تقليدية صحيحة تساوى صفراً. وهذه هى احدى المواقف التى يتم فيها انتاج متغير بدون توضيح رسمى.

٣ – اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ – انتقل إلى الدرس السادس والخمسين للاستمرار فى تسلسل التعلم.

الدرس السادس والسبعون

عبارة LINE

الوصف

ترسم عبارة LINE خطا أو صندوقاً على الشاشة، وتكوينها هو كما يلى:

```
LINE STEP x1,y1 - STEP x2,y2,color,B|BF,style
```

يتسبب جزء STEP فى رسم الاحداثيات المعطاة بالنسبة إلى أحدث احداثيات مرسومة. الأجزاء x1, y1, x2, y2 هى الاحداثيات القطبية التى يرسم بينها الخط من x1, y1 إلى x2, y2. وجزء color الاختيارى هو اللون الذى يرسم به الخط، ويمكن استخدام الجزء B أو الجزء BF، ويحدد B أن المطلوب رسم صندوق أما BF فيحدد أن المطلوب هو ملء الصندوق. وفى أى من الحالتين يرسم صندوق بين محور الاحداثيات المعطاة، وجزء style هو غطاء من 16 بت يستخدم فى رسم نقاط الرسم على الشاشة، ويستخدم فى عمل شكل للخط line styling حيث تقرأ عبارة البت فى شكل معين من اليسار إلى اليمين، 1 بت ترسم نقاط اما 0 بت فلا تفعل ذلك. ويستخدم style مع الخطوط والصناديق المعتادة وليس له أى تأثير على الصناديق المملوءة.

التطبيقات

تستخدم عبارة LINE إذا كانت امكانيات الرسومات متاحة فى رسم الخطوط، ويمكن أن تستخدم فى رسم خطوط الرسومات ورسومات الأعمدة (وذلك بخيار BF) ورسومات التقديم لتوضيح البيانات والأفكار.

عملية تقليدية

هذه العملية توضح استخدام عبارة LINE فى رسم رسومات أعمدة، يقبل البرنامج مدخلات عن بيانات الرسم ويرسم رسم أعمدة على الشاشة. استمر إذا كانت لديك امكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                           <Untitled>
'This program demonstrates the LINE statement.
DIM A%(10)

FOR Cnt = 1 TO 10
  READ t%
  A%(Cnt) = t%
NEXT

'Set the screen for medium resolution
SCREEN 1
CLS

PRINT "Your Graph: "
'Set the X axis

LINE (0, 180)-(280, 180)

'Set the starting position for the first graph
```

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                           <Untitled>
x1 = 0: y1 = 180: x2 = 20: y2 = 0

FOR Cnt = 1 TO 10
    t% = A%(Cnt)
    'Compute the height of the bar
    y2 = t% * 2
    'Draw the box
    LINE (x1 + 5, y1)-(x2, y2), 2, BF
    'Move both x-coordinates to the right
    x1 = x1 + 20: x2 = x2 + 20
NEXT


DATA 13,25,44,65,32,22,8,5,71,8

                            Immediate
Main  <Untitled>   Context: Program not running              00001:001
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة LINE.

٣ – ارجع إلى البرنامج واحفظه كملف نصي تحت اسم LINE.BAS مع اخلاء الشاشة.

٤ – انتقل إلى الدرس الخامس والخمسين للاستمرار في تسلسل التعلم.

الدرس السابع والسبعون

عبارتا LINE INPUT و LINE INPUT#

الوصف

عبارة LINE INPUT : تقبل عبارة LINE INPUT مدخلات من لوحة المفاتيح. وتكوينها هو كما يلى :

```
LINE INPUT prompt string; string variable
```

جزء prompt string يشبه سلسلة الملقن فى عبارة INPUT ويوضع بين علامتى تنصيص مزدوجتين. وجزء string variable هو مقصد البيانات التى يتم ادخالها . وتتوفر امكانيات التنقيح الكاملة مثل عبارة INPUT تماماً وتنتهى المدخلات مع عودة العربة. ولمعرفة التفاصيل عن رموز التنقيح ارجع إلى الدرس الثانى والستين. ولا تطبع عبارة LINE INPUT علامة استفهام (الا إذا كانت جزءاً من سلسلة الملقن). لاحظ أنه يوجد متغير واحد فى عبارة LINE INPUT لقبول المدخلات. وتقرأ كل البيانات التى يتم ادخالها فى هذا المتغير حتى تحدث حركة عودة العربة.

عبارة LINE INPUT# : تقرأ عبارة LINE INPUT# سطراً من ملف تتابعى معين. وتكوينها هو كما يلى :

```
LINE INPUT #filenum, string var
```

جزء filenum هو رقم الملف المحدد له فى عبارة OPEN. وجزء string var هو مقصد البيانات المقروءة. وتقرأ البيانات حتى تحدث حركة عودة العربة. وعبارة LINE INPUT# لها متغير واحد فقط للبيانات.

التطبيقات

عبارة LINE INPUT : عبارة LINE INPUT تكون مفيدة عندما يراد قراءة سلسلة من الرموز. وميزة عبارة LINE INPUT على عبارة INPUT هى أن الرموز المحددة تقرأ كذلك كجزء من البيانات. وفيما يلى بعض الأمثلة :

مثال١

```
LINE INPUT "Enter text"; TextIn$
```

يمكن أن تكون المدخلات كما يلى :

لاحظ فى هذا المثال استخدام الفراغات والفاصلة. فى عبارة INPUT يمكن للفاصلة أن تنهى عنصر بيانات.

مثال٢

```
LINE INPUT "?": MoreText$
```

مثال٣

```
LINE INPUT: StillMore$
```

عبارة #LINE INPUT : تقارن عبارة #LINE INPUT مع عبارة LINE INPUT وتستخدم فى قراءة بيانات محددة من ملف فى متغير سلسلة. وفيما يلى بعض الأمثلة :

مثال١

```
OPEN "Journal.Dat" FOR INPUT AS #2
...
LINE INPUT #2. JEntry$
PRINT JEntry$
```

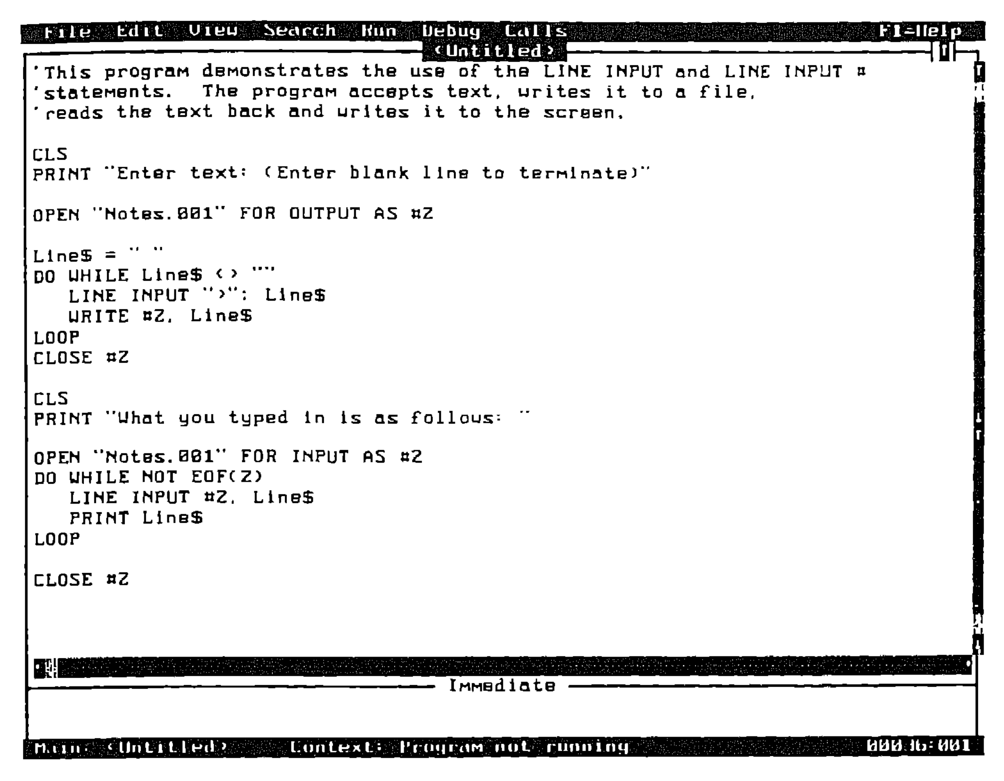
مثال٢

```
LINE INPUT #3. Line$
```

عملية تقليدية

هذه العملية توضح استخدام عبارات LINE INPUT و #LINE INPUT. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :




```
'This program demonstrates the use of the LINE INPUT and LINE INPUT #
'statements.  The program accepts text, writes it to a file,
'reads the text back and writes it to the screen.

CLS
PRINT "Enter text: (Enter blank line to terminate)"

OPEN "Notes.001" FOR OUTPUT AS #2

Line$ = " "
DO WHILE Line$ <> ""
   LINE INPUT ">": Line$
   WRITE #2, Line$
LOOP
CLOSE #2

CLS
PRINT "What you typed in is as follows: "

OPEN "Notes.001" FOR INPUT AS #2
DO WHILE NOT EOF(2)
   LINE INPUT #2, Line$
   PRINT Line$
LOOP

CLOSE #2
```

٢ – نفذ البرنامج. لاحظ استخدام عبارات LINE INPUT و LINE INPUT# فى البرنامج.
اكتب النص التالى، وعند انتهاء كل سطر اضغط على مفتاح الادخال وعندما تصل إلى
نهاية الاختيار ادخل سطراً فارغاً.

```
Enter text: (Enter blank line to terminate)
>THE PROMISE OF WORLD PEACE:
>A STATEMENT OF THE UNIVERSAL HOUSE OF JUSTICE TO THE PEOPLES OF THE WORLD.
>OCTOBER 1985.
>The Great Peace towards which people of goodwill throughout the centuries
>have inclined their hearts, of which seers and poets for countless generations
>have expressed their vision, and for which from age to age the sacred
>scriptures of mankind have constantly held the promise, is now at long last
>within the reach of the nations.  For the first time in history it is possible
>for everyone to view the entire planet, with all its myriad diversified
>peoples, in one perspective.  World peace is not only possible but
>inevitable.  It is the next stage in the evolution of this planet, in the
>words of one great thinker, "the planetization of mankind."
>
```

٣ – بعد الضغط على مفتاح الادخال مرة أخرى لادخال سطراً فارغاً تبين الشاشة ما يلى :

```
What you typed in is as follows:
 THE PROMISE OF WORLD PEACE "
"A STATEMENT OF THE UNIVERSAL HOUSE OF JUSTICE TO THE PEOPLES OF THE WORLD."
"OCTOBER 1985."
"The Great Peace towards which people of goodwill throughout the centuries"
"have inclined their hearts, of which seers and poets for countless generations"
"have expressed their vision, and for which from age to age the sacred"
 scriptures of mankind have constantly held the promise, is now at long last"
"within the reach of the nations.  For the first time in history it is possible"
 for everyone to view the entire planet, with all its myriad diversified"
 peoples, in one perspective.  World peace is not only possible but"
"inevitable.  It is the next stage in the evolution of this planet, in the "
"words of one great thinker, 'the planetization of mankind.'"
""



Press any key to continue
```

٤ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٥ – انتقل إلى الدرس المائة والثالث والستين للاستمرار فى تسلسل التعلم.

الدرس الثامن والسبعون

دالــة LOC

الوصف

تعطى دالة LOC الوضع الحالى لملف محدد. وهى لها التكوين التالى :

```
LOC(filenum)
```

جزء filenum هو رقم الملف المحدد له فى عبارة OPEN. مع ملفات الاتصال العشوائى تعطى دالة LOC رقم آخر رمز تمت قراءته أو كتابته. ومع الملفات التتابعية تعطى دالة LOC موقع البايت الحالى مقسوماً على 128. ومع الملفات الثنائية تعطى دالة LOC موقع آخر بايت تمت قراءته أو كتابته. ومع وحدة COM تعطى دالة LOC عدد البايت الذى مازال موجوداً فى صف المدخلات. ولا يمكن استخدام دالة LOC مع الوحدات التالية : SCRN و KYBRD و LPTx حيث x رقم صحيح.

التطبيقات

تستخدم دالة LOC فى حفظ تتبع مكان الاتصال بالبيانات فى الملف.

وفيما يلى بعض الأمثلة :

مثال١

```
OPEN "Temp.Dat" FOR RANDOM AS #2
..
IF LOC(2) = 0 THEN
      ..
ELSE
..
```

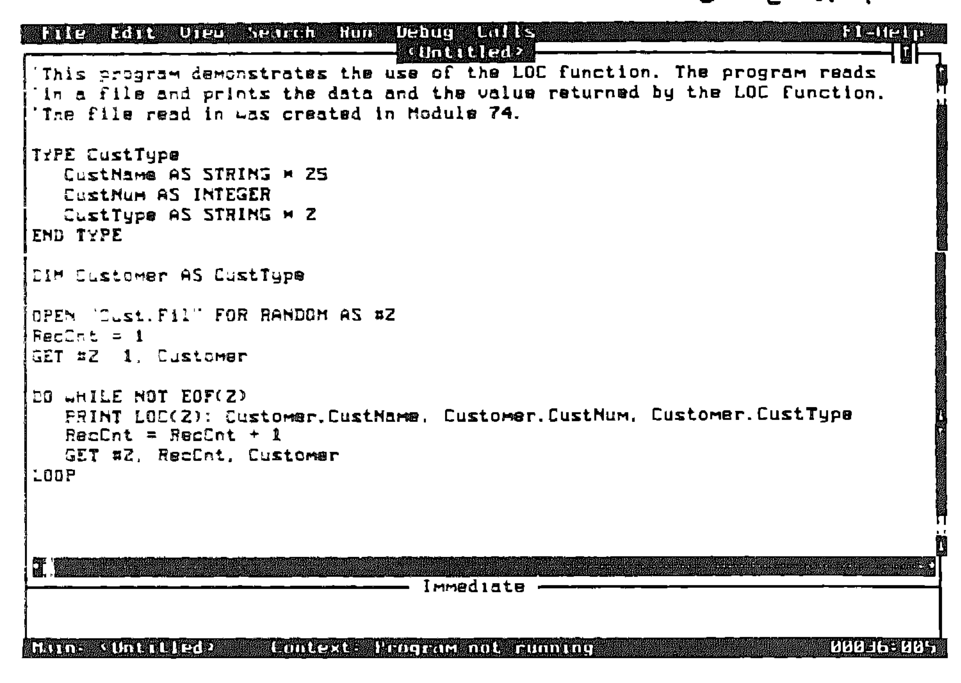
مثال٢

```
OPEN "Test.Lst" FOR RANDOM AS #1
..
WHILE LOC(1) < 3000
      ...
WEND
```

عملية تقليدية

هذه العملية توضح استخدام دالة LOC. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the use of the LOC function. The program reads
'in a file and prints the data and the value returned by the LOC function.
'The file read in was created in Module 74.

TYPE CustType
   CustName AS STRING * 25
   CustNum AS INTEGER
   CustType AS STRING * 2
END TYPE

DIM Customer AS CustType

OPEN "Cust.Fil" FOR RANDOM AS #2
RecCnt = 1
GET #2, 1, Customer

DO WHILE NOT EOF(2)
   PRINT LOC(2); Customer.CustName, Customer.CustNum, Customer.CustType
   RecCnt = RecCnt + 1
   GET #2, RecCnt, Customer
LOOP

                          Immediate

Main: <Untitled>      Context: Program not running            00036:005
```

٢ – نفذ البرنامج. لاحظ استخدام دالة LOC فى البرنامج. اضغط بعد ذلك على أى مفتاح للعودة إلى البرنامج.

```
 1 Microsoft Inc.                     333        A
 2 More Money Corp.                   1222       C
 3 Singapore Scents Co.               999        B






Press any key to continue
```

٣ - اختر New من قائمة File واختر عدم حفظ هذا البرنامج.

٤ - انتقل إلى الدرس المائة والخامس والعشرين للاستمرار فى تسلسل التعلم.

الدرس التاسع والسبعون

عبارة LOCATE

الوصف

تستخدم عبارة LOCATE فى وضع نقطة البداية على الشاشة وفى التعريف الاختيارى لخواص الشاشة. وعادة ما تستخدم مع عبارات PRINT. وتكوين عبارة LOCATE هو كمايلى:

```
LOCATE row,column,cursor,start,stop
```

والخمس مكونات فى التكوين موصوفة فى الجدول التالى :

المؤشر	المعنى	النوع
Row	الصف الذى تظهر فيه نقطة البداية.	عدد صحيح
Column	العمود الذى تظهر فيه نقطة البداية.	عدد صحيح
Cursor	حالة نقطة البداية : (1) مرئية و (0) غير مرئية.	بوليان
Start	بداية سطر الفحص scan line لمعرفة موقع نقطة البداية.	عدد صحيح
Stop	نهاية سطر الفحص scan line لمعرفة موقع نقطة البداية.	عدد صحيح

وسطر الفحص scan line هو سطر يعرف حدود نقطة البداية. وفى حالة اللون الأحادى للشاشة يمكن أن تحتل نقطة البداية حتى 14 سطر فحص من 0 إلى 13. أما فى حالة النصوص الملونة فيمكنها أن تحتل 7 من 0 إلى 6. وفيما يلى بعض الأمثلة :

```
LOCATE 1,1
```

يضع هذا المثال نقطة البداية عند الركن العلوى الأيسر للشاشة.

```
LOCATE 1,1,0
```

يضع هذا المثال نقطة البداية مثل المثال السابق مع عدم رؤية نقطة البداية.

```
LOCATE 40,10,1,0,7
```

يضع هذا المثال نقطة البداية عند العمود 40 والسطر 10 ويجعل نقطة البداية مرئية وهى تحتل خلية الرمز كلها.

التطبيقات

تستخدم LOCATE فى تحديد منطقة مخرجات على الشاشة واستخدام أمر PRINT مع LOCATE يكون قوياً ويعطيك تحكماً على موقع ظهور مخرجات البرنامج. انظر إلى عبارة LO-CATE فى برنامج العينة الموجود فى الدرس الثالث ولاحظ كيفية وضعها للصندوق وللنص داخل الصندوق.

عملية تقليدية

تكتب وتنفذ فى هذه العملية برنامجاً باستخدام LOCATE. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
CLS
Sel1$ = "Selection 1": Sel2$ = "Selection 2": Sel3$ = "Selection 3"
Sel4$ = "Quit": Sel5$ = "U -> UP / D -> DOWN / <CR> -> END"
LOCATE 2, 30: PRINT "MENU SELECTION"
LOCATE 4, 30: PRINT Sel1$
LOCATE 5, 30: PRINT Sel2$
LOCATE 6, 30: PRINT Sel3$
LOCATE 7, 30: PRINT Sel4$
LOCATE 9, 30: PRINT Sel5$

                              Immediate

Main: <Untitled>     Context: Program not running          00010:001
```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج. وتشبه المخرجات ما يلى:

```
                    MENU SELECTION

                    Selection 1
                    Selection 2
                    Selection 3
                    Quit

                    U -> UP / D -> DOWN / <CR> -> END









Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - اضغط على Alt-F واضغط على مفتاح الإدخال واكتب N لإخلاء الشاشة دون أن تحفظ البرنامج.

٥ - انتقل إلى الدرس المائة وتسعة للاستمرار فى تسلسل التعلم.

الدرس الثمانون

عبارتا LOCK و UNLOCK

الوصف

تتحكم عبارتا LOCK و UNLOCK فى الاتصال بملف مفتوح. وتكوينها هو كما يلى:

```
LOCK #filenum,rec|start TO end
UNLOCK #filenum,rec|start TO end
```

جزء filenum# فى كل من التكوينين هو رقم الملف المستخدم فى فتح الملف فى عبارة OPEN. جزء rec/start يحدد إما استخدام rec أو استخدام start. عندما تستخدم rec فيتم اغلاق أو عدم اغلاق السجل المطلوب. وعندما تستخدم start فإنها تحدد أول سجل أو بايت فى مدى السجلات أو البايت المراد اغلاقه أو عدم اغلاقه. يحدد جزء end أخر بايت أو سجل يراد اغلاقه أو عدم اغلاقه.

وعندما يكون الملف مفتوحاً فى الحالة الثنائية فتحدد أرقام السجلات مواقع البايت، وفى حالات أخرى، فإنها تشير إلى أرقام السجلات. وأقصى رقم للسجل يمكن استخدامه هو 2,147,483,647 وأقصى حجم للسجل هو 32,767 بايت.

التطبيقات

تستخدم عبارات LOCK و UNLOCK فى بيئة الشبكة فقط لتقدم أو لتمنع الاتصال بالملف أو بجزء منه للعمليات الأخرى فى البرنامج. وفيما يلى أمثلة لعبارات LOCK و UNLOCK:

مثال١

```
OPEN "Interest" FOR RANDOM AS #2
..
LOCK #2, 100
..
UNLOCK #2
```

مثال٢

```
OPEN "Acct.Pay" FOR INPUT AS #1
..
LOCK #1
..
UNLOCK #1
```

تغلق عبارة LOCK فى المثال الأول السجلات من 1 إلى 100 وتمنع عبارة UNLOCK اغلاق محتويات الملف. وفى المثال الثانى يكون الملف فى حالة تتابعية ولا تقدم عبارة LOCK مدى وذلك لأنه فى حالة الاتصال التتابعى تؤثر عبارة LOCK على الملف كله بغض النظر عن مدى السجلات المحدد.

عملية تقليدية

حيث إنه لا يمكن افتراض أنه متاح لك اتصال ببيئة شبكة كما أن مناقشة مثل هذه البيئة تقع خارج نطاق هذا الكتاب فلا يحتوى هذا القسم على مثال.

انتقل إلى الدرس المائة والتاسع والعشرين للاستمرار فى تسلسل التعلم.

الدرس الحادى والثمانون

دالــة LOF

الوصف

تعطى دالة LOF حجم الملف بالبايت، وتكوينها هو كما يلى :

```
LOF(filenum)
```

جزء filenum هو رقم الملف المحدد للملف فى عبارة OPEN، ويعود حجم الملف بالبايت بغض النظر عن الحالة التى يكون مفتوحاً بها الملف. عندما تستخدم LOF مع عبارة OPEN COM فإن القيمة التى تعود هى عدد البايت الحر الموجود فى الذاكرة الاحتياطية للمخرجات. ولايمكن استخدام دالة LOF مع الوحدات التالية : SCRN أو :KYBRD أو :CONS أو LPTx حيث x رقم صحيح.

التطبيقات

دالة LOF تكون مفيدة عندما تكون المعلومات الخاصة بحجم الملف ضرورية لتنفيذ البرنامج. تجنب الملفات الفارغة ويمثل عد عدد السجلات والتأكد من سعة القرص واختبار قيود البرنامج مواقف محددة من المواقف التى يمكن استغلال دالة LOF فيها. وفيما يلى بعض الأمثلة :

مثال١

```
PRINT "Current file size is ":LOF(1)
```

مثال٢

```
PRINT "Number of records : ":LOF(2)/RecordLen
```

مثال٣

```
IF LOF(2) > 0 THEN
    ...
ELSE
    PRINT "Empty file. Cannot process."
END IF
```

عملية تقليدية

هذه العملية توضح استخدام دالة LOF. أبدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                      F1=Help
                              <Untitled>
 This program illustrates the LOF function. The program uses LOF to
 give the file size in bytes

TYPE CustType
   CustName AS STRING * 25
   CustNum AS INTEGER
   CustType AS STRING * 2
END TYPE

DIM Customer AS CustType
CLS

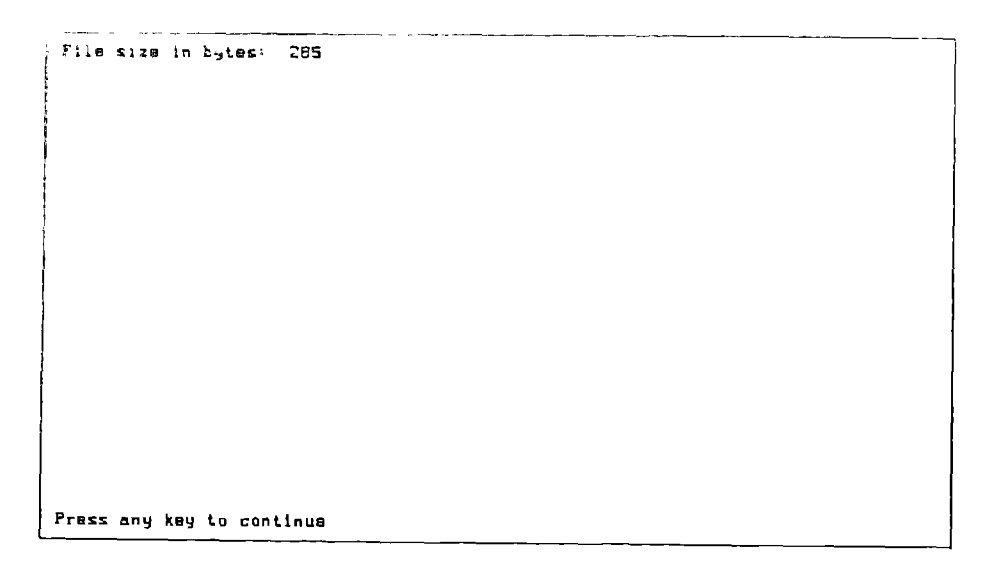
OPEN "Cust.Fil" FOR RANDOM AS #2
PRINT "File size in bytes: "; LOF(2)

CLOSE #2

                              Immediate

Main: <Untitled>     Context: Program not running              00016:009
```

٢ - نفذ البرنامج. لاحظ استخدام دالة LOF فى البرنامج.



٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - اختر New واختر عدم حفظ البرنامج.

٥ - انتقل إلى الدرس الحادى والخمسين للاستمرار فى تسلسل التعلم.

الدرس الثانى والثمانون

عبارة LOG

الوصف

تعطى عبارة LOG اللوغاريتم الطبيعى لتعبير عددى معين. وتكوينها هى كما يلى :

```
LOG(numeric expression)
```

تعيد دالة LOG اللوغاريتم الطبيعى للأساس e (وهى حوالى 2.718282). وتحسب القيمة بدقة فردية كقيمة تقليدية. وعندما يكون التعبير العددى بدقة مزدوجة فتكون قيمة LOG المحسوبة فى دقة مزدوجة كذلك. ويقوم التعبير العددى بعدد أكبر من الصفر.

التطبيقات

تستخدم عبارة LOG عندما يكون مطلوباً حساب اللوغاريتم الطبيـعى. وفيـما يلى بعض الأمثلة :

```
PRINT LOG(n)
PRINT LOG(n) / LOG(10.0)
```

يوضح هذا المثال كيفية حساب اللوغاريتم للأساس 10 باستخدام دالة LOG.

عملية تقليدية

العملية التالية توضح استخدام دالة LOG. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                                 F1=Help
                                  <Untitled>
'This program computes logarithms to any positive base.
CLS
INPUT "Enter base and x "; b, x
Anylog = LOG(x) / LOG(b)
PRINT "Log "; x; " to base "; b; " = "; Anylog

                                  Immediate

Main: <Untitled>       Context: Program not running                   00006:001
```

٢ – نفذ البرنامج. اكتب 16 كأساس و 54 على أنها x ولاحظ استخدام دالة LOG فى البرنامج.
اضغط على أى مفتاح لتعود إلى البرنامج.

```
Enter base and x ? 16,54
Log  54  to base  16  =  1.438722

Press any key to continue
```

٣ – من قائمة File اختر New واختر عدم حفظ البرنامج.

٤ – انتقل إلى الدرس المائة والحادى والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الثالث والثمانون

دالــة LPOS

الوصف

تعطى دالة LPOS الموقع الحالى لرأس الطابـع داخل الذاكرة الاحتياطية للطابع. وتكوينها هو كما يلي :

```
LPOS(n)
```

جزء n هـو رقـم الطابـع، وهـو n الموجـودة فى :LPTn، ويختـبر الطابع الموجـود فى LPT1 باستخدام LPOS(1) كما يختبر الطابع الموجود فى LPT2 باستخدام LPOS(2) وهكذا.

التطبيقات

لا تعطى دالة LPOS الموقع الفعلى الواقعى لرأس الطابع، وإنما تعطى الموقع داخل الذاكرة الاحتياطية للطابع وذلك لأن رموز الجداول لاتتسع داخل الذاكرة الاحتياطية للطابع. وفيما يلى أمثلة لدالة LPOS.

مثال١

```
LPRINT SPC(80 - LPOS(1)), "Captain"
```

مثال٢

```
IF LPOS(1) > 60 THEN LPRINT
```

يستخدم المثال الثانى دالة LPOS فى الذهاب إلى سطر جديد على الطابع.

عملية تقليدية

هذه العملية تستخدم نفس البرنامج المستخدم فى الدرس الرابع والثمانين مع تعديله لطباعة 65 حرفاً فقط فى السطر. استمر فقط إذا ما كان لديك طابع متصل بالكمبيوتر. ابدأ بتحميل بيسك السريع.

١ – حمل البرنامج المسمى LPRINT.BAS وغير فى البرنامج كما هو مبين فى القائمة التالية :

```
File  Edit  View  Search  Run  Debug  Calls                F1=Help
                          LPRINT.BAS
DIM GU%(10)
CLS
PRINT "This program will print graph values below 63"

GETINPUT:
INPUT "ENTER NEXT VALUE FOR GRAPH, 0 TO END "; I%
IF I% = 0 THEN GOTO DRAWGRAPH
GUS% = GUS% + 1: GU%(GUS%) = I%
IF (GUS% < 11) THEN GOTO GETINPUT
DRAWGRAPH:
CLS
LPRINT STRING$(10, "-"); "YOUR GRAPH"; STRING$(10, "-"): PRINT : PRINT
FOR I = 1 TO 10:
  LPRINT : LPRINT I;
  FOR I2 = 1 TO GU%(I)
    IF LPOS(1) > 65 THEN
      LPRINT
      EXIT FOR
    ELSE LPRINT "*";
    END IF
  NEXT
NEXT

                          Immediate

Main: LPRINT.BAS      Context: Program not running         00036:005
```

٢ – نفذ البرنامج ولاحظ استخدام دالة LPOS فى البرنامج.

٣ – ارجع إلى البرنامج واحفظه كملف نصى تحت اسم LPOS.BAS مع اخلاء الشاشة.

٤ – انتقل إلى الدرس المائة والحادى والستين واستمر فى تسلسل التعلم.

الدرس الرابع والثمانون

عبارتا LPRINT و LPRINT USING

الوصف

تطبع عبارتا LPRINT و LPRINT USING بيانات بواسطة الطابعة والعبارتان متشابهتان جداً مع عبارتى PRINT و PRINT USING فيما عدى أن المخرجات تتجه إلى الطابع بدلاً من اتجاهها للشاشة. وتكوينهما هو كما يلى :

```
LPRINT expression list
LPRINT USING format string: expression list
```

جزء expression list فى كل من التكوينين هو قائمة بالمتغيرات والثوابت والتعبيرات المراد طباعتها. وجزء format string يصف شكل طباعة البيانات. وللوصف الكامل لقائمة المتغيرات وسلسلة الشكل ارجع إلى الدرسين المائة وتسعة والمائة وعشرة. وتتجه المخرجات إلى الطابع الموجود فى البوابة LPT1.

التطبيقات

تستخدم عبارتا LPRINT و LPRINT USING فى طباعة بيانات على الطابع. وتفترض العبارتان أن عرض سطر الطابع 80 خانة. وفيما يلى أمثلة لعبارات LPRINT و LPRINT USING :

مثال١

```
FmtStr$ = "/   /"
LPRINT USING FmtStr$; "Money";"Money";"And";"More Money"
```

مثال٢

```
LPRINT FNLfmt$(NewStr$),NewStr$
```

مثال٣

```
FOR Cnt = 1 TO 12
    LPRINT USING "!"; Q(Cnt)
NEXT
```

عملية تقليدية

هذه العملية تستخدم البرنامج المعد فى الدرس المائة والثالث والأربعين وتطبع المخرجات على طابع. استمر إذا ما كان لديك طابع متصل بجهاز الكمبيوتر فقط. ابدأ بتحميل بيسك السريع.

١ – حمل البرنامج STRING. BAS وعدل عبارة PRINT لما هو مبين فى القائمة التالية :

```
File  Edit  View  Search  Run  Debug  Calls                      F1=Help
                              STRING.BAS
DIM GU%(10)
GETINPUT:
INPUT "ENTER NEXT VALUE FOR GRAPH, 0 TO END ", I%
IF I% = 0 THEN GOTO DRAWGRAPH
GUS% = GUS% + 1: GU%(GUS%) = I%
IF (GUS% < 11) THEN GOTO GETINPUT
DRAWGRAPH:
CLS
LPRINT STRING$(10, "-"): "YOUR GRAPH": STRING$(10, "-"): PRINT : PRINT
FOR I = 1 TO 10: LPRINT I; STRING$(GU%(I), "*"): NEXT

                              Immediate

Main: STRING.BAS      Context: Program not running            C  00010:019
```

٢ – نفذ البرنامج ولاحظ استخدام عبارة LPRINT.

٣ – اكتب 10 واضغط على مفتاح الادخال. اكتب 5 و 8 و 13 و 0 مع الضغط على مفتاح الادخال فى كل مرة يتم كتابة رقم من هذه الأرقام.

```
ENTER NEXT VALUE FOR GRAPH, 0 TO END 10
ENTER NEXT VALUE FOR GRAPH, 0 TO END 5
ENTER NEXT VALUE FOR GRAPH, 0 TO END 8
ENTER NEXT VALUE FOR GRAPH, 0 TO END 11
ENTER NEXT VALUE FOR GRAPH, 0 TO END 3
ENTER NEXT VALUE FOR GRAPH, 0 TO END 0
```

٤ – يجب أن تشبه مخرجاتك ما يلى :

```
.
----------YOUR GRAPH----------
 1 **********
 2 *****
 3 ********
 4 ***********
 5 ***
 6
 7
 8
 9
 10
```

٥ - ارجع إلى البرنامج واحفظ هذا البرنامج كملف نصى تحت اسم LPRINT. BAS مع اخلاء الشاشة.

٦ - انتقل إلى الدرس الثالث والثمانين للاستمرار فى تسلسل التعلم.

الدرس الخامس والثمانون

عبارتا LSET و RSET

الوصف

تنقل عبارتا LSET و RSET بيانات إلى ذاكرة الملف الاحتياطية من الذاكرة أو تضبط بيانات سلسلة من اليسار أو من اليمين على التوالى فى متغير سلسلة. وتكوينها هو ما يلى :

تكوين عبارة LSET :

```
LSET string var = string expression
```

جزء string var هو حقل معرف فى عبارة FIELD أو متغير سلسلة. وجزء string expression هو قيمة محددة لمتغير السلسلة. وعندما يكون متغير السلسلة أكبر من تعبير السلسلة فتضبط البيانات من الناحية اليسرى مع تكملتها بفراغات من الناحية اليمين. وعندما يكون المتغير أقل من تعبير السلسلة فتلغى الخانات الزائدة. ويجب أن تتحول القيم العددية إلى تعبيرات سلاسل باستخدام MKI$ أو MKD$ أو MKL$ أو MKS$ قبل استخدام عبارة LSET لنقل البيانات إلى ذاكرة الملف الاحتياطية أو إلى الضبط من ناحية اليسار للسلسلة.

تكوين عبارة RSET :

```
RSET string var = string expression
```

جزءا string var و string expression متطابقان تماماً مع ما هو موجود فى عبارة LSET. ويتم تشغيل البيانات بنفس الطريقة مثل عبارة LSET باستثناء أن التضبيط يحدث من الناحية اليمنى بدلاً من حدوثه من الناحية اليسرى. وجميع القيود التى تقع على LSET تقع كذلك على RSET.

التطبيقات

تستخدم عبارتا LSET و RSET فى اعداد بيانات للكتابة فى ملف اتصال عشوائى. وفيما يلى بعض الأمثلة :

عبارة LSET :

```
Size$ = SPACE$(25)
FirstName$ = "Malcolm"
LSET Size$ = FirstName$
```

يبين هذا المثال كيفية استخدام عبارة LSET فى تشكيل المتغير FirstName$ فى سلسلة طولها 25 خانة مضبوطة من ناحية اليسار. من الممكن كذلك تحديد متغير سجل لمتغير سجل آخر دون الحذف من توافقية نوع السجل. ويحدث ذلك على النحو التالى :

```
TYPE t1
  q1 A INTEGER
  ?2 AS STRING * 10
END TYPE

TYPE t2
  s1 AS STRING * 20
END TYPE

DIM r1 AS t1, r2 AS t2

LSET r1 = r2
```

ينسخ فى هذا المثال متغير السجل r2 فى متغير السجل r1.

عبارة REST :

```
LS$ = SPACE$(12)
Cuisine$ = "Mexican"
RSET LS$ = Cuisine$
```

يبين هذا المثال كيفية استخدام RSET فى تشكيل المتغير Cuisine$ فى سلسلة طولها 12 خانة مضبوطة من ناحية اليمين. وفيما يلى مثال لتحديد متغير سجل لمتغير سجل آخر له نوع مختلف عن نوعه باستخدام عبارة RSET :

```
TYPE t1
  Sample1 AS STRING * 10
END TYPE

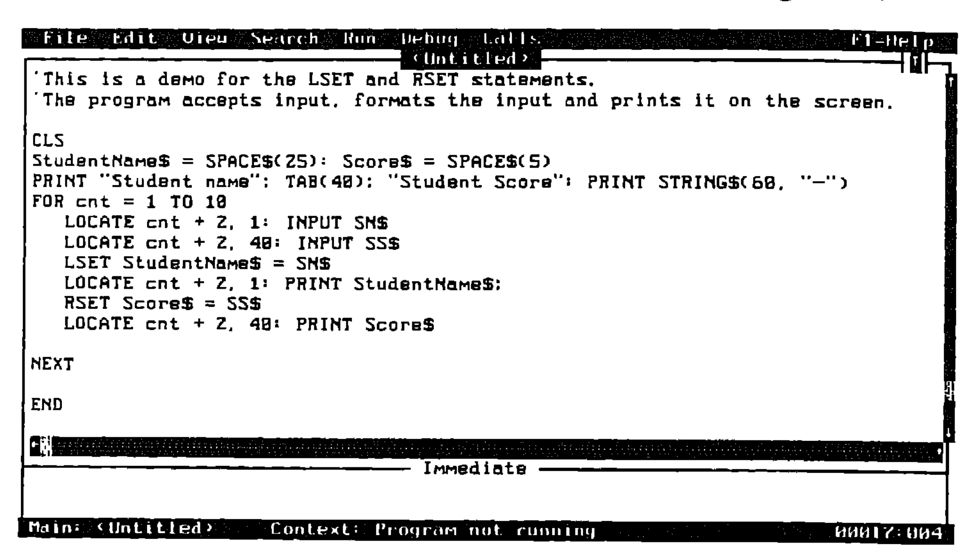
TYPE t2
  Sample2 AS STRING * 20
END TYPE

DIM Rec1 AS t1, Rec2 AS t2
RSET Rec1 = Rec2
```

عملية تقليدية

هذه العملية توضح استخدام عبارتى LSET و RSET. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This is a demo for the LSET and RSET statements.
'The program accepts input, formats the input and prints it on the screen.

CLS
StudentName$ = SPACE$(25): Score$ = SPACE$(5)
PRINT "Student name": TAB(40): "Student Score": PRINT STRING$(60, "-")
FOR cnt = 1 TO 10
   LOCATE cnt + 2, 1: INPUT SN$
   LOCATE cnt + 2, 40: INPUT SS$
   LSET StudentName$ = SN$
   LOCATE cnt + 2, 1: PRINT StudentName$;
   RSET Score$ = SS$
   LOCATE cnt + 2, 40: PRINT Score$

NEXT

END

                         Immediate

Main: <Untitled>     Context: Program not running          00017:004
```

٢ - نفذ البرنامج. لاحظ استخدام عبارات LSET و RSET فى نقل بيانات إلى الذاكرة الاحتياطية قبل طباعتها فى البرنامج. اكتب البيانات التالية مع الضغط على مفتاح الادخال بعد كل عملية ادخال. ومع اتمامك لكل سطر يعاد تشكيل البيانات. اضغط على Ctrl-Break لايقاف البرنامج.

```
Student name                              Student Score
------------------------------------------------------------
Mary Jane                                     34
Simone A.                                     55
June Ally                                    123
?
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والحادى عشر للاستمرار فى تسلسل التعلم.

الدرس السادس والثمانون

دالتا LTRIM$ و RTRIM$

الوصف

تزيل دالتا LTRIM$ و RTRIM$ الفراغات السابقة أو التالية للبيانات على التوالى وذلك من تعبير السلسلة. وتكوينها هو كما يلى :

```
LTRIM$(string exp)
RTRIM$(string exp)
```

جزء string exp فى كل من التكوينين يمثل أى تعبير سلسلة. ويمكن أن يكون تعبير السلسلة متغير سلسلة أو ثابت سلسلة أو تعبيراً ينتج عنه سلسلة. وتقبل كل من الدالتين سلاسل ثابتة الطول أو متغيرة الطول كمؤشرات لها.

التطبيقات

تستخدم دالتا LTRIM$ و RTRIM$ فى تشكيل السلاسل للطباعة أو لأى غرض آخر فى معالجة بيانات السلاسل. وفيما يلى بعض الأمثلة :

مثال١

```
Tst$ = "Please wait..   "
PRINT RTRIM$(Tst$)
```

مثال٢

```
DIM St2 AS STRING * 35
St2 = "NOT the RED button!"
PRINT LTRIM$(RTRIM$(St2))
```

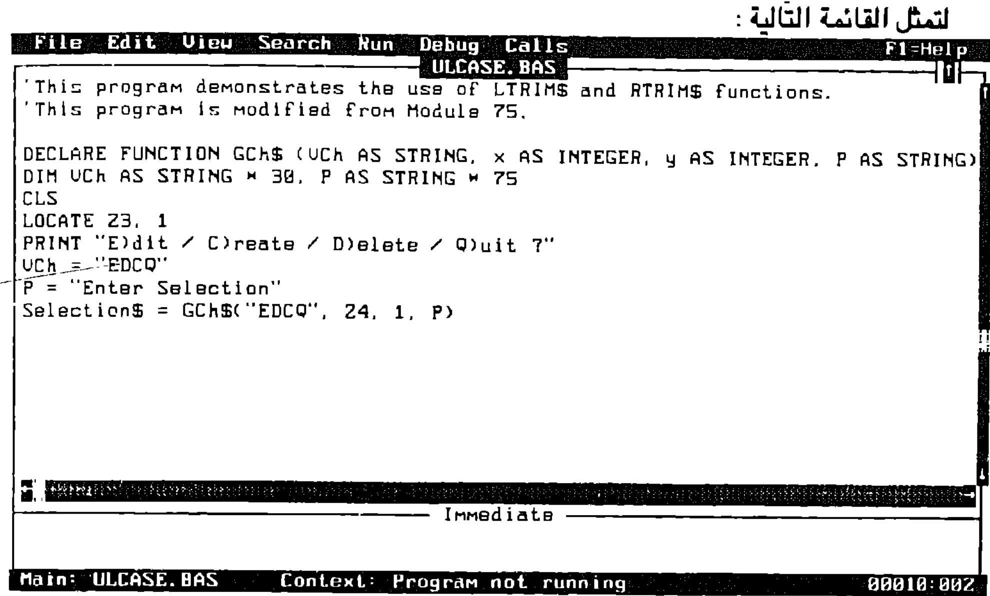
عملية تقليدية

تقوم فى هذه العملية بتعديل البرنامج المقدم فى الدرس الثانى والسبعين. وهو ULCASE.BAS، لتوضيح دوال LTRIM$ و RTRIM$. ابدأ بتحميل بيسك السريع.

١ – من قائمة File اختر Open واضغط على Tab للذهاب إلى الدليل. اختر ULCASE.BAS واضغط على مفتاح الادخال.

٢ – اضغط على Shift-F2 لتنقيح الدالة GetChar$.

٣ – لاحظ البرنامج التالى والدالة الموصوفة. استمر سطراً بسطر خلال وصفك للدالة وعدل الدالة لتمثل القائمة التالية :



```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                ULCASE.BAS
'This program demonstrates the use of LTRIM$ and RTRIM$ functions.
'This program is modified from Module 75.

DECLARE FUNCTION GCh$ (UCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
DIM UCh AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"
UCh = "EDCQ"
P = "Enter Selection"
Selection$ = GCh$("EDCQ", 24, 1, P)

                                Immediate
Main: ULCASE.BAS      Context: Program not running                    00010:002
```

٤ – اضغط على Shift-F2 لتنقيح الجزء الرئيسى. يمكن حذف الأسطر بالضغط على Ctrl-Y، احذف التعليقات الموجودة فى قمة البرنامج واضف التعليقات الجديدة كما هو مبين فى القائمة التالية. اضف عبارة DIM كما هو موضح. عدل بقية البرنامج كما هو موضح كذلك.

```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                ULCASE.BAS
'This program demonstrates the use of LTRIM$ and RTRIM$ functions.
'This program is modified from Module 75.

DECLARE FUNCTION GCh$ (UCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
DIM UCh AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"
UCh = "EDCQ"
P = "Enter Selection"
Selection$ = GCh$("EDCQ", 24, 1, P)

                                Immediate
Main: ULCASE.BAS      Context: Program not running                    00011:035
```

٥ – نفذ البرنامج ولاحظ استخدام دالة LTRIM$ ودالة RTRIM$. تظهر مدخلاتك على نفس سطر الملقن. تحل المدخلات الجديدة محل المدخلات القديمة. وعندما يتم ادخال مدخلات مناسبة فتستبدل المدخلات بالاختيار الذى تجريه.

```
E)dit / C)reate / D)elete / Q)uit ?
Enter selection?
```

٦ – اضغط على أى مفتاح للعودة إلى البرنامج.

٧ – من قائمة File اختر Save As واحذف اسم الملف ULCASE.BAS بالضغط على قضيب المسافات. اكتب LTRIM.BAS كاسم ملف جديد واضغط على مفتاح الادخال.

٨ – من قائمة File اختر New مع اخلاء الشاشة.

٩ – انتقل إلى الدرس الخامس والستين للاستمرار فى تسلسل التعلم.

الدرس السابع والثمانون

دالـة و عبارة MID$

الوصف

تعمل الكلمة المحجوزة MID$ بطريقتين طبقاً لموقع ظهورها بالنسبة إلى عبارة التحديد. فتعمل MID$ كدالة إذا ما ظهرت فى الطرف الأيمن لعبارة التحديد وتعيد جزءاً محدداً من سلسلة المؤشر. كما تعمل MID$ كعبارة إذا ما ظهرت فى الطرف الأيسر لعبارة التحديد وتستبدل جزءاً محدداً من سلسلة المؤشر بتعبير سلسلة جديد. والكلمة المحجوزة MID$ هى احدى كلمات البيسك الأكثر تعدداً للاستخدام. وتكوينها هو كما يلى :

التكوين الأول (كدالة) :

```
MID$(String expression, start, length)
```

الجـزء	الوصـف
MID$	كلمة من كلمات بيسك المحجوزة. وتعيد فى هذا التكوين الجزء المحدد من تعبير السلسلة.
string expression	سلسلة مؤشر يستخلص لها جزء مع اعادته.
start	موقع البداية للاستخلاص من تعبير السلسلة. فإذا كان موقع البداية أكبر من طول السلسلة فتعيد MID$ سلسلة فارغة.
length	مؤشر اختيارى يحدد عدد الرموز المراد استخلاصها. ويجب أن يقع بين 1 و 32,767. فاذا لم يتحدد أو إذا كانت هناك رموز أقل فى السلسلة عن الطول من البداية فتعيد MID$ كل الرموز من البداية.

التكوين الثانى (كعبارة) :

```
MID$(String variable, start, length) = String expression
```

الجـــزء	الوصــــف
MID$	كلمة من كلمات بيسك المحجوزة.
string variable	مقصد التحديد. ولا يسمح إلا بمتغيرات.
start	موقع بداية الاستبدال فى سلسلة المقصد.
length	مؤشر اختيارى يحدد عدد الرموز المراد استبدالها. فاذا لم يكن محدداً فيستخدم تعبير السلسلة كله.
string expression	سلسلة المصدر التى يعدها التحديد لمتغير سلسلة المقصد. ويمكن أن تكون متغيراً أو ثابتاً أو تعبيراً.

التطبيقات

يمكن أن تستخدم الكلمة المحجوزة MID$ فى تطبيقات معالجة سلاسل واسعة النطاق. واستخلاص جزء من سلسلة واستبدال جزء من سلسلة من أهم الاستخدامات الشائعة لها. وفيما يلى بعض الأمثلة :

```
Prompt$ = "G Men! Cheese it."
Prompt2$ = "Fuzz !"
PRINT Prompt$
MID$(Prompt$,1,6) = Prompt2$
PRINT Prompt$
```

المخرجات

```
G Men! Cheese it.
Fuzz ! Cheese it.
```

يوضح هذا المثال استخدام عبارة MID$. تستبدل السلسلة الجزئية "G MEN!" بـ Prompt 2$.

```
Prompt$ = "Here he goes again.": Prompt2$ = "What "
Prompt2$ = Prompt2$ + MID$(Prompt$,14,5) + "?"
PRINT Prompt$
PRINT Prompt2$
```

المخرجات

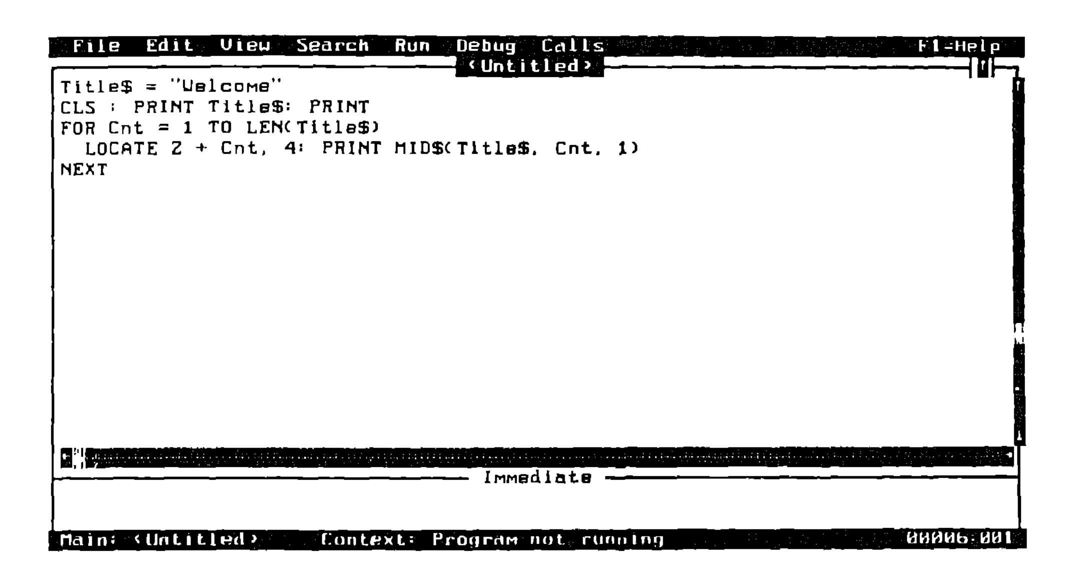
```
Here he goes again.
What again?
```

يوضح هذا المثال استخدام $MID كدالة تعيد سلسلة جزئية من تعبير سلسلة.

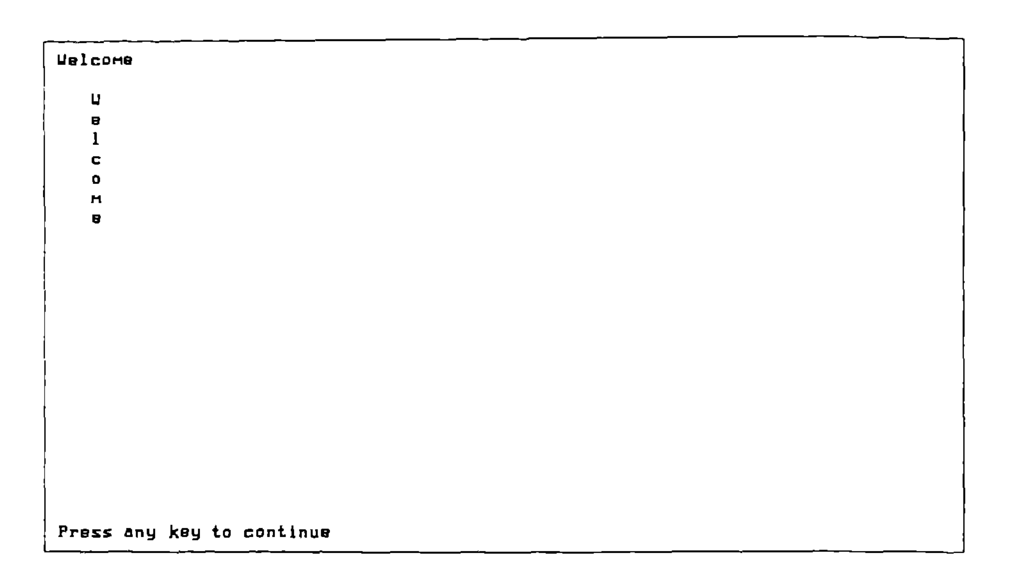
عملية تقليدية

عادة ما يطبع النص على الشاشة افقياً. وهناك طريقة مختلفة وفريدة عن هذه الطريقة وهى العرض رأسياً. هذه العملية تستخدم $MID فى برنامج لطباعة سلسلة افقياً ورأسياً. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج. لاحظ المخرجات على الشاشة واستخدام $MID فى تحقيق الطباعة الرأسية.



٣ - اضـغط على أى مـفـتـاح للعـودة إلى البـرنامج. اضـغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والعشرين للاستمرار فى تسلسل التعلم.

الدرس الثامن والثمانون

دوال MKD$ و MKI$ و MKL$ و MKS$

الوصف

تقوم الدوال MKD$ و MKI$ و MKL$ و MKS$ بتحويل قيم عددية إلى قيم سلاسل مناظرة لها. وتكوينها هو كما يلى :

```
MKI$(integer exp)
MKS$(single precision exp)
MKL$(long integer exp)
MKD$(double precision exp)
```

تحول دالة MKI$ تعبير عددى صحيح إلى سلسلة من 2 البايت.

تحول دالة MKS$ تعبير له دقة فردية إلى سلسلة من 4 بايت.

تحول دالة MKL$ تعبير عددى صحيح طويل إلى سلسلة من 4 بايت.

تحول دالة MKS$ تعبير له دقة مزدوجة إلى سلسلة من 8 بايت.

تستخدم هذه الدوال مع عبارات FIELD و PUT فى كتابة اعداد فى ملف وتحول الدوال القيم العددية إلى سلاسل بحيث يمكن تخزينها فى سلاسل معرفة فى عبارة FIELD.

التطبيقات

دوال MKD$ و MKI$ و MKL$ و MKS$ مفيدة فى تحويل القيم العددية إلى سلاسل قبل تخزينها فى ملف. وفيما يلى مثال لذلك :

```
OPEN "SalesTx.Dat" FOR RANDOM AS #3
..
FIELD #3 20 AS ItemName$, 10 AS Qty$, 12 AS SalesTx$
..
INPUT "Enter Item ";ItemName$
INPUT "Quantity   ";Qnty
INPUT "Sales tax  ";ST!
Qty$ = MKI$(Qnty): SalesTx$ = MKS$(ST!)
..
PRINT #3, ItemName$, Qty$, SalesTx$
```

يوضح المثال كيفية استخدام دالة MKI$ ودالة MKS$ فى تحويل قيم عددية صحيحة وأخرى ذات دقة فردية إلى سلاسل بحيث يمكن كتابتها فى الملف SalesTx. Dat. ويمكن أن يتحقق نفس التأثير باستخدام أنواع سجلات وملفات مرتبة لأنواع سجلات من هذه الأنواع. ويحدث ذلك عن طريق توضيح نوع سجل يعرفه المستفيد بعبارات TYPE و END TYPE وتعريف ملف يحتوى على سجلات من هذا النوع. ويوضح المثال التالى نفس العملية باستخدام عبارات TYPE و END TYPE. لاحظ بساطة الشفرة بالمقارنة بالمثال السابق.

```
TYPE  SalesTx
     ItemName AS STRING * 20
     Qty      AS STRING * 10
     STax     AS STRING * 12
END TYPE
..
DIM STRec AS SalesTx
OPEN "SalesTx.Dat" FOR RANDOM AS #3 LEN = LEN(STRec)
..
INPUT "Enter item ";STRec.ItemName
INPUT "Quantity   ";STRec.Qty
INPUT "Sales tax  ";STRec.STax
..
PRINT #3,STRec
```

عملية تقليدية

هذه العملية توضح استخدام دوال MKD$ و MKI$ و MKS$. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program illustrates the use of MKS$, MKD$, MKI$, and MKL$ functions.
'The program creates a file using these functions.

OPEN "Cust2.Fil" FOR RANDOM AS #2
FIELD #2, 25 AS CName$, 2 AS CNum$, 8 AS CreditLim$, 4 AS LastInv$

DO WHILE UCASE$(Choice$) <> "Y"
   INPUT "Enter customer name:    ": Cn$
   INPUT "Customer number    :    ": Cnumber
```

```
   INPUT "Credit limit        :    ": CrLim#
   INPUT "Last invoice        :    ": LInv!
   INPUT "Done ? (Y/N)   ": Choice$
   LSET CName$ = Cn$
   LSET CNum$ = MKI$(Cnumber)
   LSET CreditLim$ = MKD$(CrLim#)
   LSET LastInv$ = MKS$(LInv!)
   PUT #2
LOOP

CLOSE #2

RecCnt = 1

──────────────────── Immediate ────────────────────

Main: <Untitled>    Context: Program not running    00016:020
```

٢ - نفذ البرنامج. لاحظ استخدام دوال MKD$ و MKI$ و MKS$ فى البرنامج. اكتب البيانات التالية مع الضغط على مفتاح الادخال بعد كل عملية ادخال.

```
Enter customer name:    ? Mission Impossible Inc.
Customer number    :    ? 1233
Credit limit       :    ? 300000.00
Last invoice       :    ? 12000
Done ? (Y/N)   ? N
Enter customer name:    ? Last Resort Motel
Customer number    :    ? 888
Credit limit       :    ? 25000.00
Last invoice       :    ? 3000
Done ? (Y/N)   ? n
Enter customer name:    ? Income Only Corp.
Customer number    :    ? 3433
Credit limit       :    ? 1000000
Last invoice       :    ? 120000
Done ? (Y/N)   ? Y







Press any key to continue
```

٣ - ارجع إلى البرنامج واختر New دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس السادس والعشرين للاستمرار فى تسلسل التعلم.

الدرس التاسع والثمانون

دوال $MKDMBF و $MKSMBF و CVDMBF و CVSMBF

الوصف

تتعامل هذه الدوال بصفة خاصة مع أشكال ميكروسوفت الثنائية Microsoft Binary For-mat وهى طريقة لتمثيل الأعداد الحقيقية داخلياً. (لاحظ جزء MBF من الدوال)، تحـول هذه الدوال الاعـداد المخـزنة فى MBF (أشكال مـيكروسـوفت الثنائيـة) إلى أشكال IEEE (معهد المهندسين الكهربائيين والالكترونيين Institute of Electric and Electronic Engineers) والعكس. وهذا مفيد بصفة خاصة حيث إن الصيغ القديمة لبيسك ميكروسوفت تستخدم صيغة MBF فى التمـثيل الداخلى وتسـمح لك هذه الدوال باسـتـخدام ملفات اتصال عشـوائى سبق انتاجها بواسطة صيغ بيسك القديمة. وتقدم اشكال IEEE بعض مزايا اضـافيـة عن أشكال MBF فى النواحى التالية :

- تشكيل IEEE له رقم أو رقمان اضافيان فى الجزء العشرى يعطى مدى أوسع للأس.
- تشكيل IEEE له المدى التالى :

النـــوع	المـــدى
single precision	من صفر 3.37*10.^38- إلى 8.43*10^-37- من صفر 8.43*10^-37 إلى 3.37*10^38
double precision	من 1.67*10^308- إلى 4.19*10^-307- من صفر 4.19*10^-307 إلى 4.19*10^308-

تصل دقة تشكيلات الأعداد فردية الدقة إلى حوالى 7 أرقام عشرية وتصل دقة الأعداد مزدوجة الدقة إلى من 15 إلى 16 رقماً عشرياً فى تشكيل IEEE.

ويتم تحويل البرامج القديمة إلى بيسك السريع بطريقتين :

- فى ترجمة منفصلة باستخدام خيار mbf/.

- بتعديل ملفات بيانات الاتصال العشوائى من البرامج القديمة واعادة ترجمة البرنامج.

وفيما يلى تكوين دوال MKDMBF$ و MKSMBF$ و CVDMBF و CVSMBF.

```
MKDMBF$(double-precision number)
MKSMBF$(single-precision number)
CVDMBF(eight-byte string)
CVSMBF(four-byte string)
```

وفيما يلى جدول بأوصاف الدوال :

الدالـــة	الوصـــف
MKDMBF$	تحول أعداداً لها دقة مزدوجة إلى سلسلة من 8 بايت
MKSMBF$	تحول أعداداً لها دقة فردية إلى سلسلة من 4 بايت.
CVDMBF	تحول سلسلة من 8 بايت إلى عدد له دقة مزدوجة.
CVSMBF	تحول سلسلة من 4 بايت إلى عدد له دقة فردية.

التطبيقات

الاستخدام الأولى للدوال MKDMBF$ و MKSMBF$ و CVDMBF و CVSMBF هو لجعلك قادراً على استخدام ملفات اتصال عشوائى سبق انتاجها باستخدام صيغ بيسك قديمة من شركة ميكروسوفت. وفيما يلى مثال لذلك :

```
TYPE OldDat
     OD1 AS STRING * 20
     OD2 AS STRING * 4
END TYPE
DIM ODRec AS OldDat
OPEN "Account.Old" FOR RANDOM AS #2 LEN = LEN(ODRec)
...
GET #2
OD2Num! = CVSMBF(OD2)
```

عملية تقليدية

هذه العملية تصف دوال MKDMBF$ و MKSMBF$ و CVDMBF و CVSMBF.

ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the use of MKSMBF$, MKDMBF$, CVSMBF and
'CVDMBF functions. The program creates a file and reads it back.

OPEN "Cust3.Fil" FOR RANDOM AS #2
FIELD #2, 25 AS CName$, 8 AS CreditLim$, 4 AS LastInv$
CLS

DO WHILE UCASE$(Choice$) <> "Y"
   INPUT "Enter customer name:  ": Cn$
   INPUT "Credit limit        :  ": CrLim#
   INPUT "Last invoice        :  ": LInv!
   INPUT "Done ? (Y/N) ": Choice$
   LSET CName$ = Cn$
   LSET CreditLim$ = MKDMBF$(CrLim#)
   LSET LastInv$ = MKSMBF$(LInv!)
   PUT #2
LOOP

CLOSE #2

OPEN "Cust3.Fil" FOR RANDOM AS #2
FIELD #2, 25 AS CName$, 8 AS CreditLim$, 4 AS LastInv$
GET #2, 1

DO WHILE NOT EOF(2)
   PRINT CName$; CVDMBF(CreditLim$); CVSMBF(LastInv$)
   GET #2
LOOP

CLOSE #2

──────────────────────────── Immediate ────────────────────────────

<Main>  <Untitled>      Context: Program not running           00001:001
```

٢ - نفذ البرنامج ولاحظ استخدام دوال MKDMBF$ و MKSMBF$ و CVDMBF و CVSMBF. ادخل البيانات التالية :

```
Enter customer name:  ? Moonlighting Co.
Credit limit       :  ? 200000
Last invoice       :  ? 12000
Done ? (Y/N) ? n
Enter customer name:  ? Precocious Kids Inc.
Credit limit       :  ? 2000000
Last invoice       :  ? 120000
Done ? (Y/N) ? y
Moonlighting Co.          200000   12000
Precocious Kids Inc.      2000000   120000






Press any key to continue
```

٣ - ارجع إلى البرنامج واختر New دون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس المائة والتاسع عشر للاستمرار فى تسلسل التعلم.

الدرس التسعون

عبارة ..NAME.. AS

الوصف

تستخدم عبارة ..NAME..AS فى اعادة تسمية ملف على قرص، وتكوينها هو كما يلى :

```
NAME old name AS new name
```

المؤشران old name و new name عبارة عن أسماء ملفات صحيحة من DOS مع مواصفات مسار اختيارية. ويجب أن توضع الأسماء بين علامتى تنصيص، وعلى عكس أمر RENAME لاعادة التسمية من نظام DOS فيمكن استخدام عبارة ..NAME..AS فى نقل ملف من أحد الأدلة إلى دليل آخر ولكن على نفس القرص، ولا يمكن اعادة تسمية الدلائل باستخدام ..NAME.. AS .

التطبيقات

تستخدم عبارة ..NAME..AS فى اعادة تسمية ملفات اثناء تنفيذ البرنامج، وفيما يلى بعض الأمثلة :

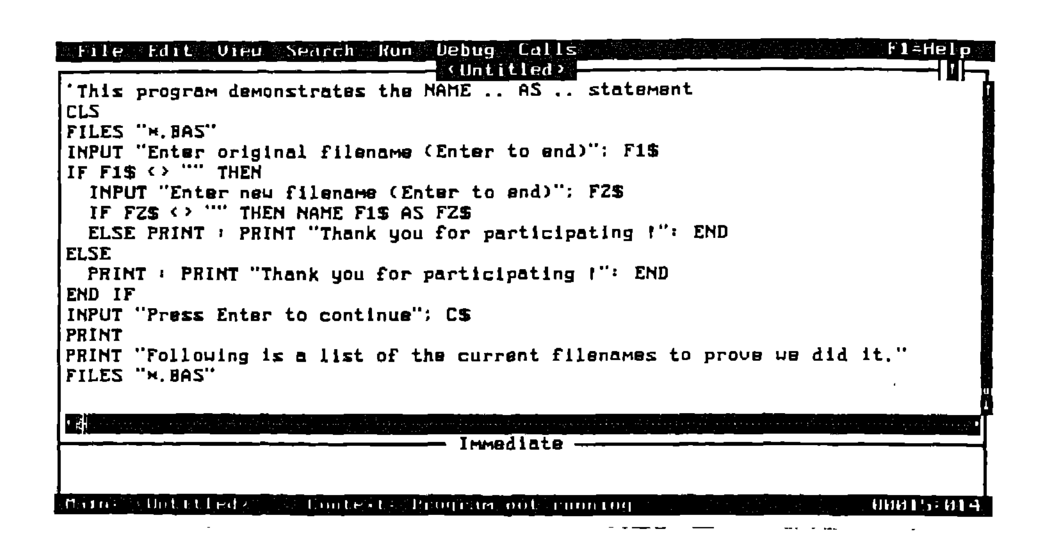
```
NAME "PCOMM.CFG" AS "PCOMM.BAK"
NAME "\GAMES\PIKMAN.EXE" AS "\GAMES\PIKMAN.OLD"
NAME "\UTIL\PCFORMAT.COM" AS "\DOS\PCFORMAT.COM"
```

يوضح المثال الأخير نقل ملف عبر دلائل باستخدام عبارة ..NAME.. AS.

عملية تقليدية

العملية التالية توضح استخدام عبارة ..NAME.. AS. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This program demonstrates the NAME .. AS .. statement
CLS
FILES "*.BAS"
INPUT "Enter original filename (Enter to end)"; F1$
IF F1$ <> "" THEN
  INPUT "Enter new filename (Enter to end)"; F2$
  IF F2$ <> "" THEN NAME F1$ AS F2$
  ELSE PRINT : PRINT "Thank you for participating !": END
ELSE
  PRINT : PRINT "Thank you for participating !": END
END IF
INPUT "Press Enter to continue"; C$
PRINT
PRINT "Following is a list of the current filenames to prove we did it."
FILES "*.BAS"

                         Immediate
```

٢ – نفذ البرنامج. لاحظ استخدام ..NAME.. AS فى البرنامج. اضغط على مفتاح الادخال لانهاء البرنامج دون اجراء أى تغيير على أسماء الملفات.

```
C:\QB
SAMPLE  .BAS      REMLINE .BAS      SORTDEMO.BAS      TORUS   .BAS
DEMO1   .BAS      DEMO2   .BAS      DEMO3   .BAS      INCH2CM .BAS
BOX     .BAS      BOX2    .BAS      PRINT   .BAS      STRING  .BAS
IFTHEN  .BAS      CASE    .BAS      ASC     .BAS      ULCASE  .BAS
LRTRIM  .BAS      ABS     .BAS      RANDOM  .BAS
 897024 Bytes free
Enter original filename (Enter to end)?

Thank you for participating !










Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس التاسع والستين للاستمرار فى تسلسل التعلم.

الدرس الحادي والتسعون

دالــة OCT$

الوصف

تعيد دالة OCT$ المكافىء الثماني لتعبير عشري، وتكوينها هو كما يلي :

```
OCT$(numeric expression)
```

يقرب التعبير العددي لأقرب قيمة صحيحة قبل التحويل. فإذا كانت القيمة المقربة خارج المدى الصحيح (من 0 إلى 32,767) فتتحول هذه القيمة إلى رقم صحيح طويل قبل تقويمها. والقيمة التي تعود تكون من نوع السلسلة ولا يمكن أن تستخدم في الحسابات.

التطبيقات

تستخدم دالة OCT$ في الحصول على التمثيل الثماني لتعبير عددي. والمبرمجون المهتمون بمثل هذا التمثيل للبيانات يجدون هذه العملية مفيدة لهم. وفيما يلي بعض الأمثلة :

```
O$ = OCT$(256*4)
PRINT "Decimal 10 is octal " OCT$(10)
```

المخرجات : الرقم العشري 10 له مكافىء ثماني 12.

عملية تقليدية

عينة البرنامج في هذه العملية توضح دالة OCT$. لاختبار ذلك ابدأ بتحميل بيسك السريع

١ – اكتب البرنامج التالي :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                <Untitled>
'This is a demonstration of OCT$ statement
CLS
FOR A% = 33 TO 128
  PRINT CHR$(A%); " "; OCT$(ASC(CHR$(A%))); "/ ";
NEXT

                               Immediate

Main: <Untitled>      Context: Program not running                00005:005
```

٢ – نفذ البرنامج ولاحظ استخدام OCT$ فى البرنامج.

```
! 41/ " 42/ # 43/ $ 44/ % 45/ & 46/ ' 47/ ( 50/ ) 51/ * 52/ + 53/ , 54/ - 55/ .
56/ / 57/ 0 60/ 1 61/ 2 62/ 3 63/ 4 64/ 5 65/ 6 66/ 7 67/ 8 70/ 9 71/ : 72/ ; 73
/ < 74/ = 75/ > 76/ ? 77/ @ 100/ A 101/ B 102/ C 103/ D 104/ E 105/ F 106/ G 107
/ H 110/ I 111/ J 112/ K 113/ L 114/ M 115/ N 116/ O 117/ P 120/ Q 121/ R 122/ S
 123/ T 124/ U 125/ V 126/ W 127/ X 130/ Y 131/ Z 132/ [ 133/ \ 134/ ] 135/ ^
136/ _ 137/ ` 140/ a 141/ b 142/ c 143/ d 144/ e 145/ f 146/ g 147/ h 150/ i 151
/ j 152/ k 153/ l 154/ m 155/ n 156/ o 157/ p 160/ q 161/ r 162/ s 163/ t 164/ u
 165/ v 166/ w 167/ x 170/ y 171/ z 172/ { 173/ | 174/ } 175/ ~ 176/ ⌂ 177/ Ç
200/

Press any key to continue
```

٣ – اضغط على أى مفتاح للعودة إلى البرنامج. من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس السابع للاستمرار فى تسلسل التعلم.

الدرس الثانى والتسعون

عبارة ON event GOSUB

الوصف

تستخدم عبارة ON event GOSUB فى اصطياد خطأ حدث وللتأثير على تنقيح البرنامج طبقاً لنتائج مثل هذه الأحداث. وتكوين عبارة ON event GOSUB هو كما يلى :

```
ON event GOSUB line number ! line label
```

يعنى الخط الرأسى ان يكون رقم السطر line number أو اسم السطر line label للبرنامج الفرعى محدداً. جزء event هو الوحدة التى توجه لحدث معين. وفيما يلى الأنواع المختلفة للأحداث.

الحـــدث	الوصـــف
TIMER (n)	يختبر وحدة TIMER ويميز متى تمر n ثانية. يقع الرقم n بين 1 و 86,400.
COM (n)	يختبر وحدة COM بالنسبة للرموز التى يتم تلقيها عند بوابة الاتصالات n. والرقم n إما أن يكون 1 أو 2.
PLAY (n)	يختبر صف الموسيقى PLAY لعدد الرموز المتروك للعب. ويحدث الحدث عندما يصبح طول الصف أقل من n.
KEY (n)	يختبر ما إذا كان المفتاح المذكور على أنه n فى عبارة KEY مضغوطاً أم لا.
PEN	يختبر ما إذا كان القلم الضوئى نشطاً أم لا.
STRING (n)	يختبر عصا الحركة وما إذا كانت مضغوطاً عليها أم لا. الرقم n هو القادح الذى يكون مضغوطاً.

يتم عمل الاصطياد وابطال عمله باستخدام ما يلى :

```
event ON            'enables event trapping
event OFF           'disables event trapping
event STOP          'suspends event trapping
```

البرنامج الفرعى الذى يستدعى بواسطة عبارة ON event GOSUB ينفذ حدث STOP لمنع اصطياد الاعادة الذاتية وينفذ حدث ON عند انتهاء البرنامج الفرعى، أى حدث يحدث بينما يكون الحدث STOP نشطاً يتم تذكرته وتشغيله عندما تنفذ عبارة الحدث ON.

التطبيقات

تستخدم عبارة ON event GOSUB فى تطبيقات البرمجة المطورة. واصطياد وتشغيل مثل هذه الأحداث يكون مفيداً جداً عندما يكون من اللازم عمل تحكم شديد على بيئة التنفيذ. فإيجاد ما إذا كانت المفاتيح مضغوطاً عليها أم لا وماذا يحدث عند احدى بوابات الاتصالات وماذا يحدث عند الوحدات الأخرى وتوجيه عبارات خلفية الموسيقى والمقدرة على استخدام هذه المعلومات لزيادة الدقة فى منطق البرنامج تمثل الاستخدام الأساسى لعبارات ON event GOSUB. ويلى بعد ذلك أمثلة لعبارات ON event GOSUB.

يكتشف أول مثال متى يقل عدد الملاحظات المتروكة فى خلفية صف الموسيقى عن 10 ثم يقفز عند ذلك إلى البرنامج الفرعى PlayContinue.

```
PLAY ON
ON PLAY(10) GOSUB PlayContinue
..
PLAY ..
PlayContinue:
     ..
     ..
     RETURN
```

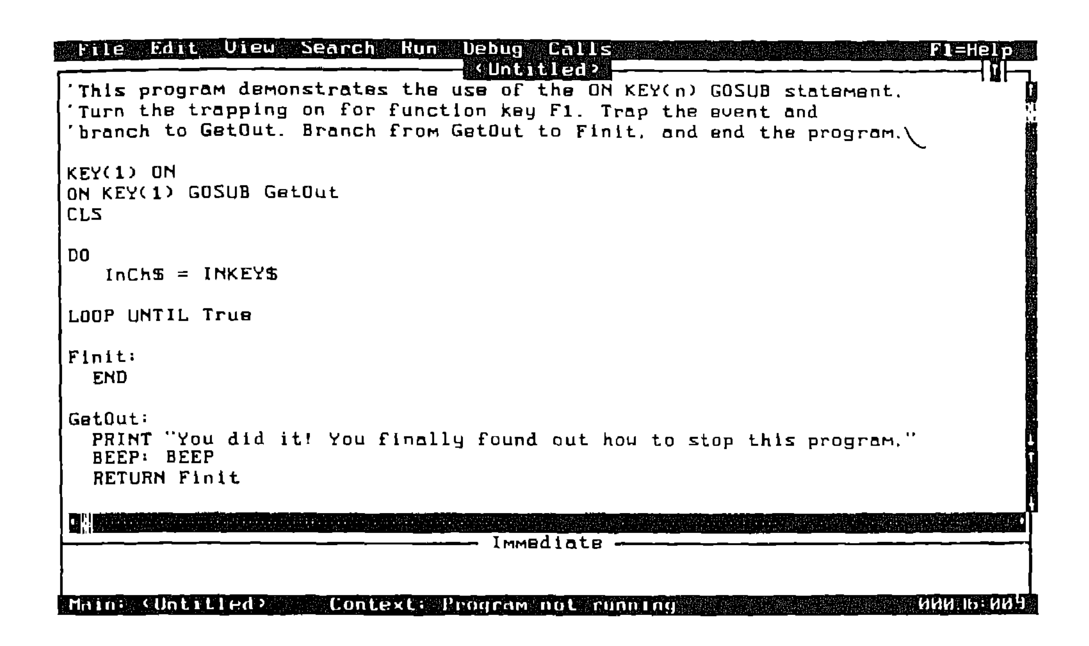
ويمكن المثال التالى من اصطياد مفتاح الوظيفة F3، اصطياد متى يكون المفتاح F3 مضغوطاً عليه، ثم يقفز بعد ذلك إلى البرنامج الفرعى AbortJob.

```
KEY (3) ON
ON KEY(3) GOSUB AbortJob
..
AbortJob:
    INPUT "Are you sure ? (Y/N) ";Yn$
    ..
    RETURN
```

عملية تقليدية

هذه العملية عبارة عن مثال لكيفية استخدام عبارة ON event GOSUB. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the use of the ON KEY(n) GOSUB statement.
'Turn the trapping on for function key F1. Trap the event and
'branch to GetOut. Branch from GetOut to Finit, and end the program.

KEY(1) ON
ON KEY(1) GOSUB GetOut
CLS

DO
   InCh$ = INKEY$

LOOP UNTIL True

Finit:
  END

GetOut:
  PRINT "You did it! You finally found out how to stop this program."
  BEEP: BEEP
  RETURN Finit

                          Immediate

Main: <Untitled>      Context: Program not running           00016:005
```

٢ – نفذ البرنامج. ولايقافه اضغط على F1. لاحظ كيفية استخدام عبارة ON event GOSUB فى البرنامج لتعريف مفتاح محدد وتشغيل هذا الحدث. اضغط على أى مفتاح للعودة إلى البرنامج.

٣ – من قائمة File اختر Save. اكتب ONEVENT. BAS كاسم للملف وحدد أن شكل الملف نصى واحفظ هذا البرنامج.

٤ – انتقل إلى الدرس الثالث والتسعين للاستمرار فى تسلسل التعلم.

الدرس الثالث والتسعون

عبارات ON.. GOTO و ON.. GOSUB

الوصف

تتسبب عبارات ON.. GOTO و ON.. GOSUB فى التفريع إلى رقم سطر محدد أو اسم سطر محدد طبقاً للقيمة المعطاة فى التعبير. وتكوينها هو كما يلى :

```
ON expression GOTO line1,line2
ON expression GOSUB line1,line2
```

جزء expression هو أى تعبير عددى ينتج عنه رقم صحيح كنتيجة له. وعندما لا تكون النتيجة رقماً صحيحاً فتقرب إلى أقرب رقم صحيح قبل تنفيذ GOTO أو GOSUB. ويتفرع البرنامج إلى line1 إذا ما كانت نتيجة التعبير 1 وإلى line2 إذا ما كانت نتيجة التعبير 2 وهكذا. وينفذ البرنامج السطر التالى عندما تكون نتيجة التعبير 0 أو أى رقم أكبر من عدد أرقام أو أسماء الأسطر الموجودة فى العبارة. وتنتج رسالة خطأ بحدوث استدعاء غير سليم لدالة عندما ينتج عن التعبير رقماً سالباً أو رقماً أكبر من 25.

الجزءين ON.. GOTO و ON.. GOSUB هما أرقام أو أسماء أسطر يتفرع إليها البرنامج طبقاً لقيمة التعبير العددى. وهناك حد ضمنى لعدد أرقام أو أسماء الأسطر الذى يتبع عبارة ON.. GOTO و ON.. GOSUB وهو 255. ويمكن أن تخلط أرقام الأسطر وأسماء الأسطر فى قائمة واحدة.

والفرق بين ON.. GOTO و ON.. GOSUB يقع فى نوع التفريع الذى ينفذ. ففى ON.. GOTO يكون التفريع شبيهاً بعبارة GOTO بدون امكانية ذاتية للعودة إلى العبارة التى تؤدى إلى التفريع. أما فى ON..GOSUB فيكون التفريع مثل عبارة GOSUB ويتوقع وجود عبارة RETURN للعودة إلى العبارة التى أدت إلى التفريع.

التطبيقات

تقدم عبارات ON.. GOTO و ON.. GOSUB طريقة قوية أخرى لعمل قرارات التفريع فى البرنامج. وهى تمثل عبارة IF.. THEN.. ELSE متعددة المستويات أو عبارة CASE STATE-

MENT وذلك فى صورة أكثر ايجازاً فقط. وتستخدم عبارات ON.. GOTO و ON.. GOSUB فى المواقف التى تكون النتائج المختبرة فيها بسيطة ولا تتطلب تكوين تفريع متعدد المستويات أكثر تطوراً. وفيما يلى بعض الأمثلة :

مثال١

```
ON Se% GOTO 120,130,150,200,1000
```

مثال٢

```
ON Fahrenheit GOTO Frozen,Freezing,VeryCold,Cold,Thawing
```

مثال٣

```
ON Result GOSUB SimpleInterest,CompoundInterest,NothingForYou
```

مثال٤

```
Input "Enter selection ";Choice%
ON Choice% GOTO FileSelect,EditRecord,NewRecord,1200
```

عملية تقليدية

هذه العملية توضح استخدام عبارة ON..GOSUB. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
CLS
PRINT : PRINT "Centigrade TO Fahrenheit TO Centigrade ..."
PRINT : INPUT "1 -> C to F / 2 -> F to C": Fc
ON Fc GOSUB CZF, FZC
END

CZF:
  INPUT "Enter temperature in Centigrade ": Cent
  PRINT : PRINT "Temperature in Centigrade "; Cent;
  PRINT "in Fahrenheit "; (9 * Cent + 160) / 5
  RETURN

FZC:
  INPUT "Enter temperature in Fahrenheit ": Farhn
  PRINT : PRINT "Temperature in Fahrenheit "; Farhn;
  PRINT "in Centigrade "; (5 * Farhn - 160) / 9
  RETURN

                              Immediate

Main: <Untitled>        Context: Program not running          00017:001
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة ON.. GOSUB فى البرنامج. أكتب 2 واضغط على مفتاح الادخال. اكتب 32 واضغط على مفتاح الادخال. اضغط على أى مفتاح للعودة إلى البرنامج.

```
Centigrade TO Fahrenheit TO Centigrade ....

1 -> C to F / 2 -> F to C? 2
Enter temperature in Fahrenheit ? 32

Temperature in Fahrenheit ' 32  in Centigrade  0











Press any key to continue
```

٣ – من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ – انتقل إلى الدرس الواحد والأربعين للاستمرار فى تسلسل التعلم.

الدرس الرابع والتسعون

عبارة OPEN

الوصف

تعد عبارة OPEN للمدخلات أو للمخرجات فى أو من ملف أو وحدة، ويجب أن يكون الملف أو الوحدة مفتوحاً قبل أى محاولة ادخال مدخلات أو الحصول على مخرجات باستخدامه. وتحدد عبارة OPEN ذاكرة احتياطية للمدخلات أو المخرجات من الملف أو الوحدة وتحدد حالة الاتصال المستخدم مع الذاكرة الاحتياطية. وهناك صيغتان لعبارة OPEN. وتكوين كل منهما هو كمايلى:

التكوين الأول :

```
OPEN file FOR mode ACCESS access lock AS # filenum LEN=reclen
```

جزء file هو تعبير سلسلة يحدد الوحدة اختيارياً واسم الملف أو اسم المسار طبقاً لاصطلاحات DOS لتسمية الملفات والمسارات. جزء FOR mode يرمز إلى نوع النشاط المراد تنفيذه على الوحدة أو الملف. والصيغ المختلفة للحالة هى كما يلى :

الحالة	المعنى
OUTPUT	مخرجات تتابعية.
INPUT	مدخلات تتابعية.
APPEND	مخرجات تتابعية. تضع مشير الملف عند نهاية الملف وتعد لاضافة مخرجات من هذه النقطة. ويوضح رقم السجل لآخر سجل.
RANDOM	اتصال عشوائى. وهذه هى الحالة التقليدية. عندما لا يوجد جزء اتصال فيحاول بيسك السريع أن يفتح الملف أو الوحدة ثلاث مرات فى الترتيب التالى : – قراءة وكتابة. – كتابة فقط. – قراءة فقط.
BINARY	اتصال بملف ثنائى. يسمح بالاتصال بالملف أو بالوحدة على مستوى البايت. يستخدم عبارات GET و PUT. وعندما لايوجد جزء ACCESS فيحاول بيسك السريع فتح الملف ثلاث مرات مثل حالة RANDOM.

جزء access فى ACCESS يعطى نوع العملية المسموح بها على الملف أو الوحدة. ويعمل ذلك فقط على وحدات تعمل تحت صيغ DOS التى تدعم استخدام الشبكات (الصيغة 3.0 وما بعدها). برنامج بدء الشبكة (SHARED. EXE) ينفذ ليسمح لأى تسهيلات اغلاق وعندما تستخدم مع صيغ DOS قديمة تظهر رسالة بأن السمات المطورة غير متاحة. وعندما يتحدد الاتصال للملف أو لوحدة مفتوحاً بالفعل ولا تتفق طريقة الاتصال الجديدة معه فتظهر رسالة بعدم امكانية الفتح. وطرق الاتصال المختلفة هى ما يلى :

الاتصال	المعنى
READ	قراءة فقط.
WRITE	كتابة فقط.
READ WRITE	قراءة وكتابة. وهذا صحيح مع حالات RANDOM و BINARY و APPEND فقط.

يستخدم جزء lock فى بيئة التشغيل المتعدد. وهذا يتحكم فى الاتصال بالملف أو الوحدة داخل النظام. وفيما يلى المواصفات المختلفة لجزء الاغلاق :

الاغلاق	المعنى
Default	عندما لا يتحدد اغلاق فيكون الملف أو الوحدة متاحاً لهذه العملية فقط. وبقية البرامج التى تنفذ فى نفس الوقت يكون لها اتصال مرفوض.
SHARED	أى برنامج فى النظام يمكنه الاتصال بهذا الملف أو هذه الوحدة.
LOCK READ	لا يمكن قراءة الملف أو الوحدة بواسطة أى عملية أخرى. ويعمل ذلك إذا لم تكن هناك مواصفة اغلاق تعطى اتصال READ للملف أو الوحدة فقط.
LOCK WRITE	لا يمكن الكتابة فى الملف أو الوحدة بواسطة أى عملية أخرى. وتطبق نفس القيود عليها مثل LOCK READ.
LOCK READ WRITE	لا يمكن القراءة أو الكتابة من وإلى الملف أو الوحدة بواسطة أى عملية أخرى. وتطبق نفس القيود عليها مثل LOCK READ و LOCK WRITE.

جزء #AS يعطى رقماً (رقم ملف filenum) كرقم للملف والذى يقع بين 1 و 255. ويصاحب هذا الرقم الملف طالما أن الملف مفتوح ويتم الاتصال بالملف باستخدام هذا الرقم. جزء = LEN يعطى حجم السجل (طول السجل reclen) بعدد الرموز. بالنسبة للفات الاتصال التتابعى يكون حجم السجل التقليدى 512 وبالنسبة للفات الاتصال المباشر يكون حجم السجل التقليدى 128. وهناك حد على طول السجل وهو 32,767 بايت. عندما تكون حالة الملف ثنائية BINARY فيهمل هذا الجزء. يمكن أن تختلف أحجام السجلات داخل الملف التتابعى ولا يحتاج جزء طول السجل أن يتوافق مع أى حجم سجل فردى آخر.

التكوين الثانى :

```
OPEN mode,# filenum, file, reclen
```

وجزء mode فى هذا التكوين هو أحد الأجزاء التالية :

الحالـة	المعـنى
O	مخرجات تتابعية.
I	مدخلات تتابعية.
R	اتصال عشوائى.
B	ملف ثنائى.
A	اتصال تتابعى. تضع مشير الملف عند انتهاء الملف ورقم سجل لآخر سجل.

وتناظر المكونات الأخرى العناصر الموجودة فى التكوين الأول. وهذه الصيغة لعبارة OPEN لا توفر تسهيلات اتصال ومشاركة كما هو الحال فى المكون الأول وهى معدة لأغراض توافقية عمل الصيانة مع الصيغ القديمة للبيسك.

والوحدات المدعمة كجزء من جزء الملف file هى كما يلى :

KYBRD: لوحة المفاتيح، مدخلات فقط

SCRN: موجه أو شاشة، مخرجات فقط.

COMn: بوابة الاتصالات، n هو رقم البوابة، مدخلات ومخرجات.

LPTn: طابع أسطر، n هو رقم الطابع، مخرجات فقط.

CONS: شاشة، مدخلات ومخرجات.

كما يسمح كذلك بالوحدات التي يعرفها المستفيد. ولتفاصيل أكثر عن مثل هذه الوحدات أفحص دليل DOS.

التطبيقات

عبارة OPEN تكون في الأساس متمركزة مع تشغيل الملف في البرنامج. أي برنامج يتطلب مللفات على قرص خارجي أو ينقل معلومات إلى وحدات يجب أن يستخدم عبارة OPEN. وتستخدم الملفات في كل التطبيقات تقريباً. وفيما يلي بعض الأمثلة :

التكوين الأول :

```
OPEN "Client.Dat" FOR RANDOM AS #3
```

يفتح هذا المثال ملفاً له الاسم "Client.Dat" للاتصال العشوائي مع الملف رقم 3.

```
OPEN "Welcome.Txt" FOR OUTPUT AS #9
OPEN "Scratch.Pad" FOR INPUT AS #11
OPEN "Temp.Buf" FOR APPEND AS #1
OPEN "Network.Dir" FOR RANDOM ACCESS READ LOCK WRITE AS #100
```

التكوين الثاني :

```
OPEN "O" , 1, "Address"
```

يفتح هذا المثال ملفاً له الاسم "Address" لمخرجات مع الملف رقم 1.

```
OPEN "R", 10, "Client.Ndx"
OPEN "B", 11, "Register.Exe"
```

عملية تقليدية

هذه العملية توضح استخدام عبارة OPEN. ابدأ بتحميل بيسك السريع.

١ - أكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the OPEN statement.

CLS
ON ERROR GOTO FileError
PRINT : PRINT "Demonstration of the OPEN statement ..."
FILES "???.*"
PRINT : INPUT "Enter filename to view: "; FileName$

IF FileName$ <> "" THEN
  OPEN FileName$ FOR INPUT AS #1
  PRINT : PRINT "Listing of file "; FileName$

  DO WHILE NOT EOF(1)
    LINE INPUT #1, InLine$
    PRINT InLine$
  LOOP

END IF

END

FileError:
  PRINT "Abnormal (?!) program termination.", ERR, ERL

                              Immediate

Main: <Untitled>      Context: Program not running              00036:004
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة OPEN فى البرنامج. تبين الشاشة دليلاً مختلفاً به ملفات مختلفة. اكتب اسم الملف الذى حفظته اثناء تسلسل التعلم. لاحظ أن الملف يجب أن يكون فى شكل ASCII لكى يعمل البرنامج بطريقة صحيحة. لقد استخدمنا الملف ASC.BAS فى هذا المثال.

```
Demonstration of the OPEN statement ...
C:\QB
        .     <DIR>          ..   <DIR> BC       .EXE      QB       .EXE
LIB     .EXE     QB       .HLP    QB       .LIB      QB       .QLB
QB      .PIF     QB       .BI     BOX      .BAS      BOX      .OBJ
BOX     .EXE     EXE      .MAP    ASC      .BAS      ABS      .BAS
UP      .TXT     002              003                004
005              W        .DAT    SUB      .BAS
 2869248 Bytes free

Enter filename to view: ? ASC.BAS

Listing of file ASC.BAS
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
  Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%



Press any key to continue
```

٣ - اضـغط على أى مـفـتـاح للعـودة إلى البـرنامج. اخـتـر Save من قـائمـة File واكـتب OPEN. BAS كاسم للملف وحدد أن شكل الملف نصى واحفظ هذا البرنامج.

٤ - اختر New من قائمة File ثم اكتب N لإخلاء الشاشة.

٥ - انتقل إلى الدرس الخامس والأربعين للاستمرار فى تسلسل التعلم.

الدرس الخامس والتسعون

عبارتا OPEN COM و COM

الوصف

عبارة OPEN COM : تفتح عبارة OPEN COM قناة اتصالات مع وضع قيمة ابتدائية لها بالنسبة للمدخلات والمخرجات. وتكوين عبارة OPEN COM هو كما يلى :

```
OPEN "COMn: oplistl oplist2" FOR mode AS #filenum LEN=Reclen
```

جزء COMn هو اسم قناة الاتصالات المستخدمة مثل : COM1 أو : COM2.

جزء oplist1 له الشكل التالى : سرعة، تعادل، بيانات، توقف. ويجب أن تتبع المحتويات هذا الترتيب المحدد وعندما تكون المحتويات محذوفة فيجب أن تستخدم الفواصل فى تحديد الأماكن. ويصف الجدول التالى جزء oplist1.

الخيــار	الوصــف
Speed	عدد البت فى الثانية كمعدل لنقل البيانات (معدل بود).
Parity	N = لاشيء و E = زوجى و 0 = فردى و S = سرعة و M = علامة.
Data	عدد بت البيانات، 5 أو 6 أو 7 أو 8.
Stop	عدد بت التوقف، 1 أو 1.5 أو 2.

وفيما يلى قائمة بمحتويات جزء m. oplist2 فى القائمة لها قيمة تقليدية 1000.

الخيـار	الوصــف
ASC	يفتح الوحدة فى حالة ASCII. تتسع Tab وتجبر على عودة العربة عند انتهاء كل سطر، Ctrl-Z بدلاً من EOF ويستخدم اتفاق XON/ XOFF.
BIN	يفتح الوحدة فى الحالة الثنائية وهى الحالة التقليدية.
CDm	تضع سطر Data Carrier Detect فى التعليق بعد m ميللى ثانية.
CSm	تضع سطر Clear To Send فى التعليق بعد m ميللى ثانية.
LF	تسمح بطباعة ملف اتصالات على الطابع. ويجبر رمز تغذية السطر على الظهور بعد عودة العربة.
OPm	تحدد طول العبارة التى تنتظر لعملية فتح ناجحة.
RBn	تحدد حجم الذاكرة الاحتياطية المستقبلة بأنها n بايت. القيمة التقليدية هى 512 بايت.
RS	تضغط اكتشاف Request To Send.
TBn	تحدد حجم الذاكرة الاحتياطية للنقل بأنها n بايت. القيمة التقليدية هى 512 بايت.

وجزء mode هو أحد ما يلى والقيمة التقليدية هى RANDOM :

الحالـة	الوصــف
OUTPUT	مخرجات تتابعية.
INPUT	مدخلات تتابعية.
RANDOM	حالة اتصال عشوائى.

جزء filenum # هو رقم الملف المستخدم فى فتح الوحدة. وجزء LEN يعطى طول السجل. عندما تفتح الوحدة على أنها RANDOM فإن LEN يساوى الذاكرة الاحتياطية للاتصال العشوائى. والقيمة التقليدية لجزء LEN هى 128 بايت.

وعندما تنفذ عبارة OPEN COM فإنها تؤدى الأشياء التالية :

- تحدد ذاكرات احتياطية وتمكن من الازعاجات interrupts.
- تحدد أن DTR مرتفع.
- عندما يكون خيار OP أو DS غير صفرى فتنتظر حتى يصبح DTR مرتفعاً أو حتى يكون هناك تعليق فإذا كان هناك تعليق فتفشل OPEN COM.
- تحدد أن RTS مرتفع إذا كان خيار RS محذوفاً.
- عندما يكون خيار OP أو CD غير صفرى فتنتظر حتى يصبح DTR مرتفعاً أو حتى يكون هناك تعليق فإذا كان هناك تعليق فتفشل OPEN COM.
- عندما تفشل عبارة OPEN COM فيلغى تحديد الذاكرات الاحتياطية وتلغى مقدرة الازعاجات interrupts وتحمى اسطر التحكم.

عبارة COM : تمكن عبارة COM أو لا تمكن أو توقف من اصطياد الأحداث على الوحدة COMn. وتكوينها هو كما يلى :

```
COM(n) ON
COM(n) OFF
COM(n) STOP
```

جزء n فى الثلاث عبارات كلها هو رقم وحدة اتصالات مثل 1 أو 2. وتمكن عبارة COM(n) ON من اصطياد الأحداث لبوابة اتصالات محددة. والأحداث التى تحدث عند بوابة الاتصالات تكتشف وتمرر إلى المقطع الخاص بمعالجتها. وتلغى COM(n) OFF من مقدرة اصطياد الأحداث لبوابة محددة. الأحداث التى تحدث عند البوابة يتم تذكرتها وتشغيلها بعد تنفيذ تسلسل COM(n) ON.

ويجب تنشيط اصطياد الأحداث بعبارة ON COM (n) GOSUB قبل أن يمكن استخدام أى عبارة من عبارات COM (n).

التطبيقات

تستخدم عبارات OPEN COM و COM (n) ON, OFF و STOP فى فتح قناة اتصالات ونقل واستقبال بيانات على وحدة COM. وفيما يلى أمثلة لعبارات COM و COM (n).

```
OPEN "COM1: 300,E,7,1,ASC" AS #2

COM(1) ON
ON COM(1) GOSUB GetCharCom
..
OPEN "COM1: 1200,N,8,1,BIN" AS #1
..
GetCharCom:
     GET #1, ..
     ..
     RETURN
```

عملية تقليدية

حيث إن المعلومات عن اجراءات الاتصالات لا يمكن افتراضها وأن الموضوع يقع خارج مدى هذا الكتاب فلن يحتوى هذا القسم على مثال.

انتقل إلى الدرس السابع والستين للاستمرار فى تسلسل التعلم.

الدرس السادس والتسعون

عبارة OPTION BASE

الوصف

تعرف عبارة OPTION BASE الحد السفلى التقليدى لمنظومة، وتكوينها هو كما يلى :

```
OPTION BASE n
```

جزء n اما أن يكون 1 أو 0. وهذا يعرف الحد السفلى لكل المنظومات الموجودة فى البرنامج وتستخدم عبارة OPTION BASE مرة واحدة فقط فى أحد الأجزاء وتظهر على مستوى كتابة الجزء فقط. وتستخدم عبارة OPTION BASE قبل تحديد أبعاد أى منظومة. وعندما تتصل البرامج ببعضها البعض فيكتسب البرنامج الذى يتم التوصيل إليه اعداد عبارة OPTION BASE للبرنامج الذى يتم توصيله به.

التطبيقات

عبارة OPTION BASE غير ضرورية فى بيسك السريع. وعبارة DIM لها مقدرة على تحديد حدود سفلى للمنظومات اثناء تحديد الأبعاد وليس عليها قيود عبارة OPTION BASE (وهى 0 و 1). ومع ذلك ففيما يلى بعض الأمثلة بغرض التوضيح.

مثال١

```
OPTION BASE 1
DIM Q(12)
```

مثال٢

```
OPTION BASE 0
DIM WorkLst(10,10)
```

عملية تقليدية

هذه العملية توضح استخدام عبارة OPTION BASE. ابدأ بتحميل بيسك السريع.

١ – اختر Open وحمل البرنامج REDIM.BAS.

٢ – عدل البرنامج ليتفق مع القائمة التالية :

```
 File  Edit  View  Search  Run  Debug  Calls                        F1=Help
                               <Untitled>
'The following program is the program from Module 115, modified to
'demonstrate the OPTION BASE statement.

OPTION BASE 1
Max = 15
DIM A(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A(Cnt)
  NEXT
  RETURN

FindMinMax:
 MinVal = A(1): MaxVal = A(1)
 FOR Cnt = 2 TO Max
   IF MinVal > A(Cnt) THEN
      MinVal = A(Cnt)
   END IF
   IF MaxVal < A(Cnt) THEN
      MaxVal = A(Cnt)
   END IF
 NEXT
 RETURN


Start:
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "First pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  READ Max
  REDIM A(Max)
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "Second pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,80,67,54,10
DATA 10
DATA 8,89,76,54,23,32,12,4,33,54

                               Immediate

Main: <Untitled>       Context: Program not running              00054:001
```

٣ – نفذ البرنامج. لاحظ استخدام عبارة OPTION BASE فى البرنامج.

```
First pass
Minimum of array:  1        Maximum of array:  656
Second pass
Minimum of array:  4        Maximum of array:  89





Press any key to continue
```

٤ – ارجع إلى البرنامج واختر New دون أن تحفظ البرنامج.

٥ – انتقل إلى الدرس الحادى والسبعين للاستمرار فى تسلسل التعلم.

الدرس السابع والتسعون

عبارة PAINT

الوصف

تملأ عبارة PAINT مناطق رسومات بلون معين أو بنمط معين. وتكوينها كما يلي :

```
PAINT STEP (x,y),paint,border,background
```

جزء STEP الاختياري يستخدم فى تحديد أن الاحداثيات المعطاة نسبية إلى موضع الشاشة الحالى. جزء (x,y) هو احداثي نقطة الرسم والذى يقع داخل المنطقة المراد تلوينها. ويجب أن تقع الاحداثيات المعطاة داخل أو خارج المنطقة المراد تلوينها وليس على حدودها. يمكن أن يكون جزء paint من النوع العددى أو السلسلة. وعندما يكون عددياً فيجب أن تكون قيمة خاصية اللون من النوع الصحيح. أما إذا كان سلسلة فلا تكسو عبارة PAINT والتى تملأ المنطقة بنمط معين بدلاً من لون واحد. جزء border هو اللون المستخدم فى تعريف حدود الشكل وعندما يرسم لون لحدود فيتوقف التلوين. وجزء background هو قيمة سلسلة تحدد الخلفية التى يجب تركها أثناء دهان الشكل.

الخلفية : تلون الخلفية بنمط معين بدلاً من لون واحد. وطريقة وصف هذا النمط هى كمايلى:

١ - ارسم النمط على شبكة من 8 أعمدة ويمكن أن يصل عدد صفوفها إلى 64.

٢ - ترجم كل صف إلى سلسلة من 0 و 1 واكتب ذلك.

٣ - حول السلسلة من 0 و 1 إلى قيم سادسة عشرية.

٤ - انتج سلسلة بدمج نتيجة دالة CHR$ للقيم السادسة عشرية التى تم الوصول إليها فى الخطوة السابقة.

٥ - لون الفراغ الموجود داخل الشكل مستخدماً خيار tile من عبارة PAINT.

التطبيقات

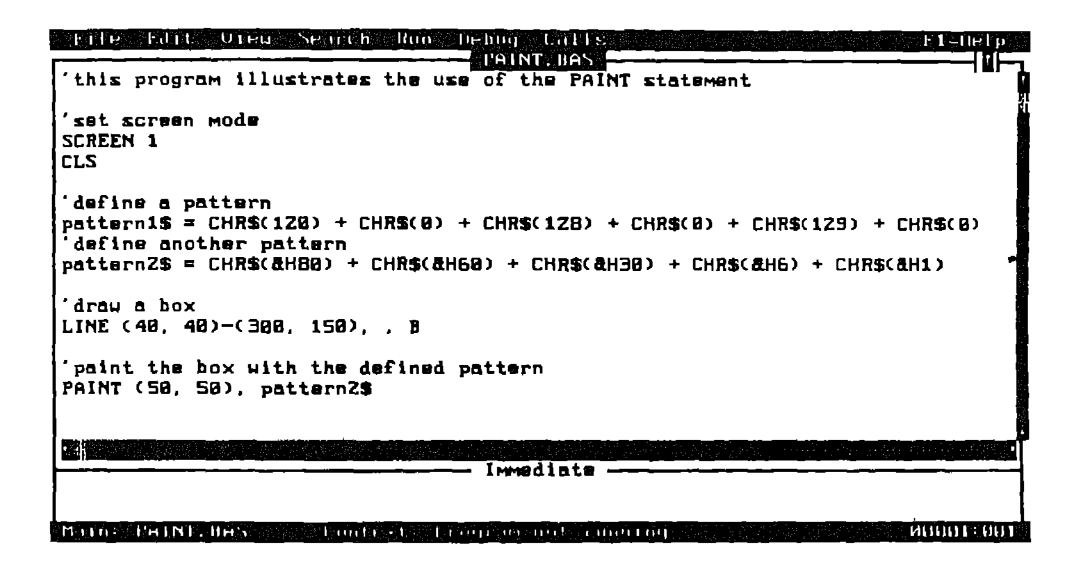
تستخدم عبارة PAINT فى تلوين أشكال الرسومات. وهذا مفيد بالنسبة للعديد من التطبيقات مثل الألعاب والتوضيحات والتقديمات. وفيما يلى مثال لعبارة PAINT.

```
SCREEN 1
COLOR 1,2
CLS
DRAW ..
CIRCLE (100,120),30
PAINT (100,120),3
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PAINT. استمر إذا ما كانت لديك امكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالي :



```
'this program illustrates the use of the PAINT statement

'set screen mode
SCREEN 1
CLS

'define a pattern
pattern1$ = CHR$(120) + CHR$(0) + CHR$(128) + CHR$(0) + CHR$(129) + CHR$(0)
'define another pattern
pattern2$ = CHR$(&H80) + CHR$(&H60) + CHR$(&H30) + CHR$(&H6) + CHR$(&H1)

'draw a box
LINE (40, 40)-(300, 150), , B

'paint the box with the defined pattern
PAINT (50, 50), pattern2$
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة PAINT فى البرنامج.

٣ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٤ – انتقل إلى الدرس المائة وثمانية للاستمرار فى تسلسل التعلم.

الدرس الثامن والتسعون

عبارتا PALETTE و PALETTE USING

الوصف

تغير عبارة PALETTE الألوان فى مجموعة الألوان، ومجموعة الألوان عبارة عن مجموعة الألوان المستخدمة حالياً فى حالة شاشة معينة، وتغير عبارة PALETTE الألوان مستبدلة إياها بمجموعة ألوان جديدة، وتكوينها هو كما يلى :

```
PALETTE attr,color
PALETTE USING arrayname(index)
```

جزء attr هو الخاصية المراد تغييرها، وجزء color هو اللون الذى يحل محل اللون الحالى لهذه الخاصية وهو تعبير عددى صحيح. ويقدم جزء USING ميزة اضافية فى أنه يستخدم منظومة أرقام صحيحة أو منظومة أرقام صحيحة طويلة فى استبدال مدى كامل للخواص attrib-utes، جزء arrayname هو اسم المنظومة وجزء index هو دليل المنظومة التى يبدأ من عندها الاستبدال.

وتعمل عبارة PALETTE مع النظم التى بها بطاقات EGA أو VGA أو MCGA فقط.

التطبيقات

تستخدم عبارة PALETTE فى رسم مجموعة من قيم الألوان بخاصية مستخدمة حالياً فى حالة شاشة معينة. وتستخدم حالة الشاشة مجموعة من الألوان فى العرض وبالرغم مما إذا كان يسمح النظام بألوان مختلفة فعبارة PALETTE يمكن أن تستخدم فى اختيار قيم ألوان للخواص معطية بذلك تحكماً أكثر فى ألوان العرض. عندما تستخدم عبارة PALETTE تتغير كل الألوان المعروضة حالياً على الفور إلى مجموعة الألوان الجديدة وتستخدم المخرجات التالية مجموعة الألوان الجديدة. ويساعد جزء USING فى اعداد مدى كامل للخواص مع عبارة PALETTE واحدة، وكل المحتويات السالبة باستثناء 1- تكون قيم ألوان غير صحيحة فى المنظومة المستخدمة فى هذا الغرض، ويقوم 1- فى المنظومة بترك الخاصية دون تغيير. استخدام عبارة COLOR بعد عبارة PALETTE يضع قيماً تقليدية لألوان الخواص. وفيما يلى بعض أمثلة لعبارة PALETTE.

مثال١

```
SCREEN 8
PALETTE 1,4
..
```

مثال٢

```
PALETTE 4,2
..
```

مثال٣

```
DIM P(15)
FOR C = 1 TO 15
    P(C) = C
NEXT
..
PALETTE USING P(0)
```

يستخدم آخر مثال منظومة أعداد صحيحة فى وضع مجموعة الألوان الجديدة لقيم الخاصية. لاحظ أن المنظومة لها 15 عنصراً، وهذا هو أقل حجم لمنظومة تستخدم فى هذا الفرض نظراً لأنه مسموح بحد أدنى 16 لوناً فى حالة بطاقة EGA وبطاقة VGA. ومع امكانية بطاقة VGA وبطاقة MCGA تستخدم منظومة اعداد صحيحة طويلة لأن قيم الألوان أكبر لهذه الحالات.

فى حالة VGA نحسب الألوان بطريقة مختلفة فاللون عبارة عن قيمة لدرجات مختلفة من الألوان الأزرق والأحمر والأخضر فى الشاشة وفى المطبع. والصيغة هى ما يلى :

```
Color = 65536 * blue + 256 * green + red
```

حيث blue و green و red هى قيم لشدة اللون تقع بين 0 و 63.

عملية تقليدية

حيث إنه لا يمكن افتراض انك لديك امكانية بطاقة EGA أو بطاقة VGA أو بطاقة MCGA فلا يحتوى هذا القسم على مثال. ويجب أن يوجد المزيد من المعلومات عن امكانية الألوان فى نظامك فى دليل المستفيد الخاص بالجهاز نفسه.

انتقل إلى الدرس المائة والثامن والخمسين للاستمرار فى تسلسل التعلم.

الدرس التاسع والتسعون

عبارة PCOPY

الوصف

تنسخ عبارة PCOPY صفحة من الشاشة فى مكان آخر. وتكوينها هو كما يلى :

```
PCOPY n1,n2
```

جزء n1 هو الصفحة المراد نقلها وجزء n2 هو الصفحة المراد النقل اليها. ولمناقشة عدد الصفحات المرئية المتاحة فى حالات مختلفة للشاشة ارجع إلى الدرس الرابع والعشرين.

التطبيقات

يمكن استخدام عبارة PCOPY فى عديد من التطبيقات والتى تكون مخرجات الشاشة فيها أقصى ما يمكن وتكون سرعة المخرجات حاسمة. يمكن عرض احدى صفحات العرض أثناء اعداد صفحة أخرى. وتطبيقات الرسومات هى اختيار حتمى كما يمكن لتطبيقات النصوص الموجهة للحالة text-mode-oriented أن تتمتع من زيادة سرعة مخرجات الشاشة. وفيما يلى أمثلة لعبارة PCOPY:

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                        <Untitled>
'This program demonstrates the use of the PCOPY statement.

COLOR 2, 1
CLS
'Write a bunch of asterisks
FOR i = 1 TO 100
  PRINT "*";
NEXT

'Copy the current page to page 1
PCOPY 0, 1

CLS
'Write a bunch of dashes
FOR i = 1 TO 1000
  PRINT "-";
NEXT

'Copy this to page 2
PCOPY 0, 2
```

```
'Copy back first from page 1, pause, then from page 2
FOR i = 1 TO 3
  PCOPY 1, 0
  FOR t = 1 TO 500: NEXT
  PCOPY 2, 0
  FOR t = 1 TO 500: NEXT
NEXT
                         Immediate
Main: <Untitled>      Context: Program not running          00036:005
```

عملية تقليدية

هذه العملية تستخدم البرنامج المقدم فى قسم التطبيقات من هذا الدرس. استمر إذا كانت لديك إمكانيات عمل رسومات ملونة لدعم ذلك. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج الموجود فى قسم التطبيقات فى هذا الدرس.

٢ - نفذ البرنامج. لاحظ المخرجات واستخدام عبارات PCOPY فى البرنامج.

٣ - احفظ البرنامج كملف نصى تحت اسم PCOPY. BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس المائة وواحد للاستمرار فى تسلسل التعلم.

الدرس المائة

دالة PEEK وعبارة POKE

الوصف

تعمل دالة PEEK وعبارة POKE مع ذاكرة الكمبيوتر وتكمل كل منهما الأخرى.

دالة PEEK : تعيد دالة PEEK البايت عند موقع محدد للذاكرة، وتكوينها هو كما يلى :

```
PEEK(address)
```

جزء address هو قيمة تقع فى المدى من 0 إلى 65,535. تعيد دالة PEEK قيمة عددية صحيحة تقع فى المدى من 0 إلى 255. والقطاع الذى يكون فيه الفرع هو القطاع المحدد بواسطة عبارة DEF SEG. وفى غياب عبارة DEF SEG يستخدم قطاع بيانات بيسك السريع.

عبارة POKE : تكتب عبارة POKE بيانات فى موقع محدد من الذاكرة، وتكوينها كمايلى :

```
POKE address, byte
```

جزء address هو تعبير عددى يقع فى المدى من 0 إلى 65,535 وجزء byte هو تعبير عددى يقع فى المدى من 0 إلى 255. والعنوان يكون الفرع بنفس الطريقة مثل دالة PEEK.

التطبيقات

تستخدم PEEK فى قراءة معلومات معينة من ذاكرة الكمبيوتر وفى اتخاذ قرارات مبنية على ذلك. وتستخدم POKE فى الكتابة فى مواقع الذاكرة مباشرة. واحد التطبيقات هو الرسومات والرسومات المتحركة مثل المباريات. وفيما يلى مثال لذلك.

```
'This program demonstrates how the POKE function is used for simple animation.
'On non IBM PC's, or full compatibles this program may not work.
'Set the segment to the monochrome screen starting address.
'Use &HB800 for color monitors
DEF SEG = &HB000
CLS
FOR x = 1 TO 10
'POKE the asterisk; the multiplication with 160 causes the char. to move downwards
  POKE x * 160, 42
'wait
  FOR 1 = 1 TO 300: NEXT
'POKE a space where the asterisk was
  POKE x * 160, 32
NEXT x
'Reset the default segment
DEF SEG
```

المخرجات من هذا المثال تبين نجمة تتحرك من قمة الشاشة لأسفل.

عملية تقليدية

هذه العملية توضح استخدام PEEK و POKE. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                 <Untitled>
'This is a demonstration of the PEEK statement. The program writes the
'alphabet at the top of the screen, finds the letter Q and changes it
'to another character.
'On non-IBM PC's, or full compatibles this program may not work.
'Set the default segment to the monochrome screen starting address.
'Use &HB800 for color monitors.
DEF SEG = &HB000
CLS
PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

FOR x = 1 TO 2048

'PEEK at what is there: convert that to a character.
  IF CHR$(PEEK(x)) = "Q" THEN
'POKE the ASCII character 42 (an asterisk) where Q was, beep, and leave.
    POKE x, 42
    BEEP
    EXIT FOR
  END IF
NEXT

'Reset the default segment
DEF SEG

                                Immediate

Main: <Untitled>        Context: Program not running                00036:001
```

٢ - نفذ البرنامج. لاحظ استخدام PEEK و POKE فى البرنامج.

```
ABCDEFGHIJKLMNOP*RSTUVWXYZ




Press any key to continue
```

٣ - ارجع إلى البـرنامـج. احـفظ البـرنامج كـملف نصى تحت اسم POKE.BAS مع اخـلاء الشاشة.

٤ - انتقل إلى الدرس التاسع للاستمرار فى تسلسل التعلم.

الدرس المائة وواحد

دالة PEN

الوصف

تعطى دالة PEN إحداثيات القلم الضوئى، وتكوينها هو كما يلى :

```
PEN(n)
```

حيث n قيمة تقع فى المدى من 0 إلى 9. ويعطى الجدول التالى القيم التى تعود لكل قيمة من قيم n.

N	القيمة التى تعود
0	1- إذا تحرك القلم لاسفل من عند أخر استخدام لدالة PEN وإلا فان القيمة تكون 0.
1	احداثى x لنقطة الرسم لأخر استخدام للقلم.
2	احداثى y لنقطة الرسم لأخر استخدام للقلم.
3	1- إذا كانت حركة القلم الحالية لاسفل و 0 إذا كانت لأعلى.
4	أخر احداثى x لنقطة الرسم يكون صحيحاً.
5	أخر احداثى y لنقطة الرسم يكون صحيحاً.
6	وضع صف الشاشة لأخر قلم مستخدم.
7	وضع عمود الشاشة لأخر قلم مستخدم
8	وضع صف الرمز لأخر قلم مستخدم.
9	وضع عمود الرمز لأخر قلم مستخدم.

التطبيقات

تستخدم دالة PEN فى قراءة احداثيات القلم الضوئى وهى محددة لهذه الوحدة فقط. ولا تعمل دالة PEN مع مشغل الفأرة لأن مشغل الفأرة يستخدم نفس استدعاءات BIOS مثل دالة PEN. وفيما يلى بعض الأمثلة :

```
PEN ON
PPos = PEN(3)
IF PPos < 0 THEN PRINT "Pen down" ELSE PRINT "Pen up"

PEN ON
FOR PP = 0 TO 9
     PRINT PEN(PP)
NEXT PP
```

يطبع أخر مثال كل القيم الممكنة التى تعود بواسطة دالة PEN. لاحظ كذلك عبارات PEN ON فى المثال وهذا ضرورى لأن القلم الضوئى يبدأ فى التوقف ويجب أن يعاد قبل أن يمكن استخدام دالة PEN. وتظهر رسالة خطأ بأن هناك استدعاء دالة خطأ إذا لم يحدث ذلك.

عملية تقليدية

حيث إنه لا يمكن افتراض استخدام وإتاحة القلم الضوئى فلا يقدم هذا القسم مثالاً.

انتقل إلى الدرس المائة واثنين للاستمرار فى تسلسل التعلم.

الدرس المائة واثنان

عبارات PEN ON و PEN OFF و PEN STOP

الوصف

تقوم عبارات PEN ON و PEN OFF و PEN STOP بتمكين وعدم تمكين والغاء تصيد حدث القلم الضوئى. وأساسيات استخدام كل عبارات الاحداث ON و OFF و STOP هى نفس الأساسيات. وتتعامل هذه المجموعة من العبارات مع القلم الضوئى. للمزيد من المعلومات عن اصطياد الأحداث أرجع إلى الدرس الثانى والتسعين. وتكوين عبارات PEN ON و PEN OFF و PEN STOP هو كما يلى :

```
PEN ON
PEN OFF
PEN STOP
```

تمكن عبارة PEN ON من اصطياد الأحداث للقلم الضوئى. وتستخدم العبارة PEN (n) GOSUB/GOTO فى التاثير على أحد الأحداث عند اصطيادها.

وتلغى PEN OFF مقدرة اصطياد الأحداث للقلم الضوئى. والاحداث التى تحدث بعد تنفيذ هذه العبارة لا يمكن تذكرها ولا يمكن تشغيلها بعبارات PEN ON تأتى بعد ذلك. وتوقف عبارة PEN STOP اصطياد الأحداث للقلم الضوئى. والأحداث التى تحدث بعد تنفيذ هذه العبارة يمكن تذكرها ويمكن تشغيلها بعد تنفيذ تسلسل من عبارات PEN ON.

التطبيقات

تطبيقات PEN ON و PEN OFF و PEN STOP محدودة ومحددة لاستخدام القلم الضوئى. ويغلق القلم الضوئى وينشط اصطياد الأحداث بعبارة ON PEN (n). يجب أن تنفذ عبارة PEN ON قبل أن يمكن استخدام دالة PEN.

انتقل إلى الدرس المائة والثامن والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة وثلاثة

دالـة وعبارة PLAY

الوصف

الكلمة المحجوزة PLAY يمكن أن تستخدم بطريقتين، كدالة وكعبارة. كدالة فإنها تعيد عدد النوت الموسيقية الموجودة فى الصف إذا كانت هناك موسيقى فى الخلفية وإلا فتكون نتيجتها صفراً. وكعبارة فإنها تلعب موسيقى كما هو محدد بسلسلة المؤشر. وتكوينها فى كل من الحالتين هو :

تكوين الدالة :

```
PLAY(n)
```

هذه هى دالة PLAY. ويكون n هو مؤشر صورى يمكن أن يكون أى قيمة موسيقية. تعيد PLAY عدد النوت الموسيقية المتروكة فى صف موسيقى الخلفية فى المتغير n.

تكوين العبارة :

```
PLAY string expression
```

هذه هى عبارة PLAY. يحتوى جزء string - expression على أوامر للموسيقى. ويصف الجدول التالى الأوامر المسموح بها :

الأمـــر	الأثـــر
>	يزيد هذا من الثمانية octave بمقدار 1. وأعلى ثمانية هى 6.
<	يقلل هذا من الثمانية بمقدار 1. وأقل ثمانية هى 0.
o number	يحدد هذا الثمانية والمدى من 0 إلى 6. ويرقم الثمانية من C إلى B أى ثمانية واحدة تصاعدية CDEDEFAGB encompases والقيمة التقليدية للثمانية هى 1.
N number	يلعب النوتة الموسيقية المعطاة بالرقم. الرقم يقع بين 0 و 84 (وتكون كل النوت الموسيقية فى سبعة ثمانيات). عندما يكون الرقم 0 فهذا يعنى راحة.

الأثـــر	الأمـر
يلعب نوتة موسيقية بين A و G كما هو محدد، ويمكن أن يتبع النوتة # أو + لتعنى حاداً sharp أو - لتعنى مسطحاً flat.	A - G
يحدد مدى استمرار النوت الموسيقية. ويقع الرقم فى المدى من 1 إلى 64 حيث 1 يمثل نوتة كاملة و 4 تمثل ربع نوتة وهكذا. عندما تتطلب نوتة معينة طولاً مختلفاً فيتبع النوتة طولها. مثال ذلك G12 تكافىء L12G.	L number
تحدد أن الموسيقى معتادة. كل نوتة تلعب 7/8 من الطول المحدد بواسطة L.	MN
تحدد أن الموسيقى متسقة. كل نوتة تلعب الطول المحدد بواسطة L.	ML
تحدد أن الموسيقى متقطعة. كل نوتة تلعب 3/4 الطول المحدد بواسطة L.	MS
تحدد توقفاً لحظياً فى الموسيقى. يقع الرقم بين 1 و 64 مناظراً لطول النوتة المحدد بواسطة L.	P number
تحدد السرعة للموسيقى. ويقع العدد بين 32 و 255 وله قيمة تقليدية 120. والسرعة هى عدد ربع النوت التى تلعب فى دقيقة واحدة.	T number
تحدد أن الموسيقى تلعب فى الأمامية. ويظهر صوت كل نوتة بعد انتهاء أخر نوتة. وهذا هو الوضع التقليدى.	MF
تحدد أن الموسيقى تلعب فى الخلفية. ويستمر تنفيذ البرنامج اثناء لعب النوتة. ويمكن لعب 32 نوتة كحد أقصى فى الخلفية فى نفس الوقت. ويتسبب ذلك فى لعب النوتة 3/2 من الطول المحدد بواسطة L مضروباً فى أعداد السرعة بواسطة T. يمكن استخدام فترات متعددة بعد النوتة. وتضيف كل فترة طولاً مساوياً 1/2 طول الفترة السابقة للنوتة. (وهذا يكافىء «النوتة المنقطة» باصطلاحات الموسيقى).	MB
تنفذ السلسلة الموجودة فى متغير السلسلة.	"X"+VARP-TRE$ (string variable)

التطبيقات

يمكن استخدام دالة PLAY فى التأكد من تقدم الاحداث فى موسيقى الخلفية واتخاذ قرارات بتنفيذ البرنامج. وتستخدم عبارة PLAY فى لعب الموسيقى. وفيما يلى بعض الأمثلة :

دالة PLAY : فيما يلى أمثلة لدالة PLAY :

```
NotesLeft% = PLAY(N%)
ON PLAY(10) GOSUB FillErUp
```

عبارة PLAY : فيما يلى أمثلة لعبارة PLAY

```
PLAY "o0ABo1CDEFG P2 GFEDCo0BA"
PLAY "o2 AA..BB..o3CC > AA..BB..o4CC"
PLAY "T180 o2 P2 P8 L8 GGG L2 E-"
St$ = "T180 o2 p2 p8 l8 ggg l2 e-"
PLAY "X" + VARPTR$(St$)
```

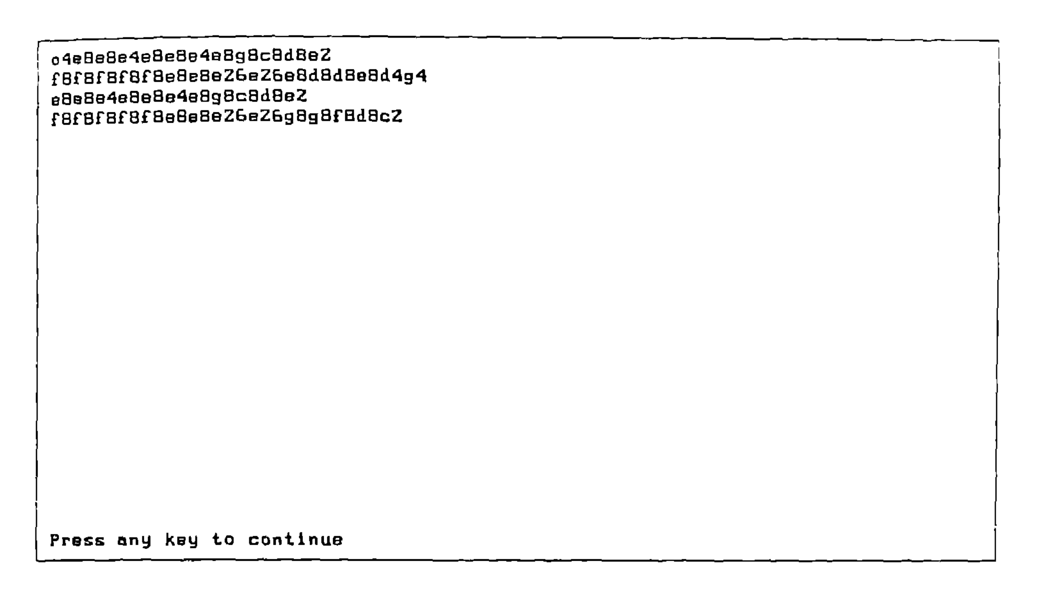
عملية تقليدية

العملية التالية توضح عبارة PLAY. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the PLAY statement
CLS
KEY OFF
PLAY "MBL16T155"
FOR I = 1 TO 4
  READ M$
  PRINT M$
  PLAY M$
NEXT I
KEY ON
END
DATA "o4e8e8e4e8e8e4e8g8c8d8eZ
DATA "f8f8f8f8f8e8e8eZ6eZ6e8d8d8e8d4g4
DATA "e8e8e4e8e8e4e8g8c8d8eZ
DATA "f8f8f8f8f8e8e8eZ6eZ6g8g8f8d8cZ
                          Immediate
Main: <Untitled>     Context: Program not running           00001:017
```

٢ - نفذ البرنامج. لاحظ الموسيقى من عبارة PLAY. لاحظ استخدام عبارة PLAY في البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.



٣ - من قائمة File اختر Save. أكتب PLAY.BAS كاسم للملف واحفظ هذا البرنامج كملف نصى.

٤ - انتقل إلى الدرس المائة وأربعة للاستمرار فى تسلسل التعلم.

الدرس المائة وأربعة

عبارات PLAY ON و PLAY OFF و PLAY STOP

الوصف

تستخدم عبارات PLAY ON و PLAY OFF و PLAY STOP فى التحكم فى اصطياد الأحداث فى البرنامج. وتمكن عبارة PLAY ON من اصطياد الأحداث وتلغى عبارة PLAY OFF إمكانية اصطياد الأحداث وتوقف عبارة PLAY STOP من اصطياد الأحداث. وتستخدم هذه العبارات بالاتصال مع عبارة ON PLAY. وتكوينها هو كما يلى :

```
PLAY ON
PLAY OFF
PLAY STOP
```

تستخدم عبارة PLAY ON فى ايجاد المقدرة على اصطياد الأحداث بحيث إنه يميز الحدث المراد اختباره عندما يحدث. ومثال للحدث هو عدد النوت الموسيقية المتروك فى موسيقى الخلفية. وتؤخذ القرارات عند حدوث الحدث.

وتستخدم عبارة PLAY OFF فى الغاء المقدرة على اصطياد الأحداث ولا يمكن تذكرة الأحداث التى تحدث.

وتستخدم عبارة PLAY STOP فى ايقاف اصطياد الأحداث ولا ينفذ اجراء معالج الأحداث. تنفذ PLAY STOP تلقائياً داخل اجراء معالج الأحداث بحيث أن الأحداث الأكثر لاتتسبب فى الاعادة الذاتية للبرنامج. والأحداث التى تحدث بعد عبارة PLAY STOP يمكن تذكرها وتشغيلها عندما تنفذ عبارة PLAY ON.

يحدث اصطياد الأحداث لعبارة PLAY عندما تلعب الموسيقى فى الخلفية فقط.

التطبيقات

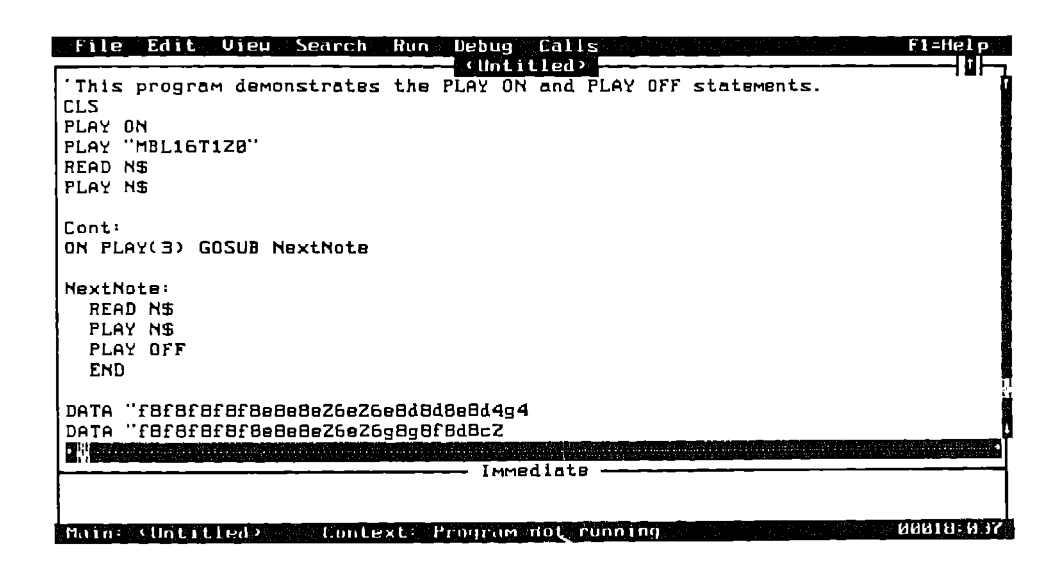
تستخدم عبارات PLAY ON و PLAY OFF و PLAY STOP عندما يكون مطلوباً تحكماً شديداً على حدوث الأحداث أثناء تنفيذ البرنامج. وتستخدم عبارات PLAY ON و PLAY OFF و PLAY STOP فى اصطياد الأحداث أثناء تنفيذ عبارة PLAY. وفيما يلى مثال لذلك.

```
PLAY ON
PLAY "GE GE > FD "
..
IF PLAY(10) THEN GOSUB  TenLeft
..
ON PLAY(3) GOSUB PlaySomeMore
..
PlaySomeMore:
     PLAY NewStr$
     ..
     PLAY OFF
RETURN
```

عملية تقليدية

هذه العملية توضح استخدام عبارات PLAY ON و PLAY OFF و PLAY STOP. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج. لاحظ استخدام عبارات PLAY ON و PLAY OFF و PLAY STOP فى البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

٣ - من قائمة File اختر N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثانى والتسعين للاستمرار فى تسلسل التعلم.

الدرس المائة وخمسة

دالـــة PMAP

الوصف

ترسم دالة PMAP أحداثيات معينة لاحداثيات منطقية أو واقعية مبنية على مفتاح الدالة.
وتكوينها هو كما يلى :

```
PMAP(expression.function)
```

جزء expression هو الاحداثيات المراد رسمها وجزء function هو المفتاح الذى يعد للرسم. ويبين الجدول التالى مفاتيح الدالة المسموح بها وما تفعله.

مفتاح الدالـة	مايفعلـــه
0	يرسم احداثيات منطقية لإحداثى x واقعى.
1	يرسم احداثيات منطقية لإحداثى y واقعى.
2	يرسم احداثيات واقعية لإحداثى x منطقى.
3	يرسم احداثيات واقعية لإحداثى y منطقى.

التطبيقات

دالة PMAP مفيدة جدا عندما تستخدم مع عبارة WINDOW لأن دالة PMAP يمكنها أن ترسم الاحداثيات الموجودة بنظام الاحداثيات السابقة التعريف. وفيما يلى مثال لدالـة PMAP :

```
SCREEN 1
WINDOW SCREEN (10,50)-(150,100)
x = 12: y = 1
x = PMAP(x,2)
y = PMAP(y,3)
```

```
SCREEN 1
COLOR 1,2
CLS
DRAW ..
DRAW ..
Xp = POINT[0]: Yp = POINT(1)
DRAW ..
DRAW "BM ="+VARPTR$(Xp)+",=",+VARPTR$(Yp)
DRAW ..
```

يوضح المثال كيفية استخدام دالة POINT فى حفظ الاحداثيات عند نقطة معينة أثناء انتاج الرسومات. وتستخدم عبارة DRAW الاحداثيات المحفوظة للعودة إلى هناك والاستمرار فى الرسم.

عملية تقليدية

المثال الموجود في الدرس الخامس والثلاثين، DRAW، هو توضيح جيد لدالة POINT. وقد حفظ البرنامج تحت اسم DRAW.BAS. راجع هذا البرنامج لرؤية احدى طرق استخدام دالة POINT.

انتقل إلى الدرس السابع والتسعين للاستمرار فى تسلسل التعلم.

الدرس المائة وستة

دالة POINT

الوصف

تعيد دالة POINT إما رقم اللون من نقطة الرسم أو احداثيات نقطة الرسم. وتكوينها هو كما يلى :

```
POINT (x,y)
POINT (number)
```

يعطى التكوين الأول قيمة اللون عند احداثيات نقطة الرسم (x,y) وعندما تكون الاحداثيات خارج المدى فتعيد POINT القيمة 1-. ويعيد التكوين الثانى احداثيات نقطة الرسم الحالية طبقاً لقيمة الرقم. ويعطى الجدول التالى القيم والاحداثيات التى تعيدها الدالة :

الرقم	الاحداثيات التى تعيدها الدالة
0	إحداثى x الواقعى.
1	إحداثى y الواقعى.
2	إحداثى x المنطقى.
3	إحداثى y المنطقى.

الاحداثى المنطقى هو الاحداثى النسبى لعبارة WINDOW النشطة حالياً.

التطبيقات

دالة POINT مفيدة فى الحصول على احداثيات نقطة الرسم الحالية وذلك أثناء رسم الصور. والوظائف المختلفة التى تسمح بها دالة POINT فى الحصول على احداثيات نقطة الرسم تحدث بأكثر من طريقة واحدة. وفيما يلى مثال لدالة POINT :

```
SCREEN 1
COLOR 1,2
CLS
DRAW ..
DRAW ..
Xp = POINT[0]: Yp = POINT(1)
DRAW ..
DRAW "BM ="+VARPTR$(Xp)+",=",+VARPTR$(Yp)
DRAW ..
```

يوضح المثال كيفية استخدام دالة POINT في حفظ الاحداثيات عند نقطة معينة أثناء انتاج الرسومات. وتستخدم عبارة DRAW الاحداثيات المحفوظة للعودة إلى هناك والاستمرار في الرسم.

عملية تقليدية

المثال الموجود في الدرس الخامس والثلاثين، DRAW، هو توضيح جيد لدالة POINT. وقد حفظ البرنامج تحت اسم DRAW.BAS. راجع هذا البرنامج لرؤية احدى طرق استخدام دالة POINT.

انتقل إلى الدرس السابع والتسعين للاستمرار في تسلسل التعلم.

الدرس المائة وسبعة

دالـة POS

الوصف

تعطى دالة POS الوضـع الحالى لنقطة البداية، وتكوينها هو كما يلى :

```
POS(column)
```

يستخدم جزء column إلا أنه يهمل، وتعيد هذه الدالة الموقع الأفقى الحالى لنقطة البداية.

التطبيقات

تستخدم دالة POS فى الحصول على معلومات عن عمود نقطة البداية وتستخدم هذه الدالة فى توفير تحكم أفضل للشاشة فى الحالة التالية ولتعزيز السطح البينى للمستفيد ببرامج تطبيقات، وفيما يلى بعض أمثلة لدالة POS.

```
IF POS(0) > 20 THEN
LOCATE CSRLIN + 1, 1
END IF
CRow = POS(0)
..
LOCATE 12, CRow+1
```

عملية تقليدية

البرنامج المستخدم فى هذه العملية هو نفس البرنامج المستخدم فى الدرس الخامس والعشرين، ويوضح البرنامج بحرية استخدام دالة POS فى تطبيق عملى، ابدأ بتحميل بيسك السريع.

١ – حمل البرنامج المستخدم فى الدرس الخامس والعشرين والمسمى CSRLIN.BAS.

٢ – أنظر إلى الدالة .GetChar بالضـغط على Shift-F2، لاحظ استـخدام دالة POS فى حفظ موقع الصف الحالى.

٣ – ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٤ – انتقل إلى الدرس الستين للاستمرار فى تسلسل التعلم.

الدرس المائة وثمانية

عبارة PRESET

الوصف

ترسم عبارة PRESET نقطة على الشاشة عند الاحداثيات المحددة. وتكوينها هو كما يلى:

```
PRESET STEP (x,y),color
```

يحدد جزء STEP أن الاحداثيات نسبية إلى الوضع الحالى وهو جزء اختيارى. ويعطى جزء (x,y) احداثيات الشاشة للنقطة المراد رسمها. ويصف جزء color خاصية اللون للنقطة واستخدام هذا الجزء اختيارى. وعندما يحذف فيكون اللون المختار هو اللون التقليدى للخلفية. أى يصبح السطر غير مرئياً.

التطبيقات

عبارة PRESET مفيدة فى رسم نقاط على الشاشة تكون مستقلة عن أشياء الرسومات الموجودة بالفعل على الشاشة. مثال ذلك أنه يمكن استخدامها فى رسم نجوم فى السماء أو فى اضافة مقطع إلى الرسم. كما يمكن استخدامها كذلك فى رسم رسومات الخطوط. والاستخدامات محدودة بالتطبيق الذى يعد له البرمجة. وفيما يلى مثال لعبارة PRESET.

```
FOR C = 1 TO 100
    PRESET(C,100), 2
NEXT
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PRESET. استمر إذا كانت لديك امكانيات رسومات ملونة تدعم ذلك فقط. ابدأ بتحميل بيسك السريع.

١ – أكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                    PRESET.BAS
'This program uses the PRESET statement.
SCREEN 1: COLOR 1, 3
CLS

FOR cnt = 1 TO 50
  PRESET (65, cnt), 2

NEXT
'Change the viewport and draw another line.
VIEW (100, 10)-(150, 30), , 1
FOR cnt = 1 TO 50
  PRESET (1, cnt), 2

NEXT

                                    Immediate

Main: PRESET.BAS        Context: PRESET.BAS                        00002:009
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة PRESET فى البرنامج.

٣ - ارجع إلى البرنامج واحفظ البرنامج كملف نصى تحت اسم PRESET.BAS.

٤ - انتقل إلى الدرس المائة واثنى عشر للاستمرار فى تسلسل التعلم.

الدرس المائة وتسعة

عبارة PRINT

الوصف

تستخدم عبارة PRINT فى عرض بيانات عددية أو بيانات سلاسل على الشاشة. وتكوينها هو كما يلى :

```
PRINT expression list|.|;
```

جزء expression list هو تسلسل من عناصر البيانات المراد طباعتها وتكون مفصولة عن بعضها البعض بواسطة فواصل أو فواصل منقوطة. وتكون البيانات العددية مشكلة طبقاً لنوعها المحدد وطباعتها. ودائماً ما توضع بيانات السلسلة بين علامتى تنصيص مزدوجتين فى عبارة PRINT. وشكل عبارة PRINT لطباعة كل من نوعى البيانات موضح أدناه.

النـوع	تعبيرالطباعـة	المخرجـات
عددى صحيح	PRINT 20	20
	PRINT - 20	-20
	PRINT 20 * 2	40
دقة فردية	PRINT 2.1E-6	0000021
	PRINT 2.1E-7	2.1E-7
(تطبع اعداد الدقة الفردية حتى 7 خانات للأرقام)		
دقة مزدوجة	PRINT 2.1D-15	.0000000000000021
	PRINT 2.1-16	2.1D-16
(تطبع أعداد الدقة المزدوجة حتى 16 خانة للأرقام)		
سلسلة	PRINT "Dead on Arrival"	Dead on Arrival

ويتحقق تشكيل المخرجات عن طريق استخدام الفراغات أو الفواصل أو الفواصل المنقوطة. وتأثير كل تشكيل مذكور فى الجدول التالى :

عبارة PRINT	المخرجات
PRINT 1, 2	1 2
(تتسبب الفاصلة فى تجزئة السطر إلى منطقتى طباعة كل منها يشغل ١٤ خانة وتطبع كل قيمة فى قائمة التعبير عند بداية المنطقة التالية)	
PRINT 1 2	1 2
PRINT 1;2	1 2
PRINT 1; : PRINT 2	1 2
(المخرجان من عبارة PRINT يوضعان على نفس السطر بسبب استخدام الفاصلة المنقوطة)	
PRINT 1 : PRINT 2	1 2
(توضع قيمة مخرجات عبارة PRINT الثانية فى السطر التالى)	

رموز التشكيل التى سبق وصفها تحدد موقع ظهور عناصر البيانات على الشاشة.

التطبيقات

الاستخدام المناسب لعبارة PRINT مع رموز التشكيل المناسبة يعد وسيلة قوية. وفى برنامج العينة الموجود فى الدرس الثالث يستخدم أمر PRINT فى رسم مستطيل وإخراج الرسالة داخل هذا المستطيل. وفيما يلى أمثلة أخرى لاستخدام عبارة PRINT.

```
PRINT "New baby !", "WOW !!"
```

وتشبه المخرجات مايلى :

```
New baby !      WOW !!
PRINT SQR(2)
```

المخرجات :

```
1.414214
FOR Cnt = 1 TO 40: PRINT "*";: NEXT Cnt
```

المخرجات :

```
........................................
```

عملية تقليدية

البرنامج التالى يوضح استخدام عبارة PRINT. يشتمل متغير السلسلة على رموز تشكيل يستخدمها البرنامج فى زيادة اضاءة نص معين. ولمحاولة ذلك ابدأ بتحميل بيسك السريع.

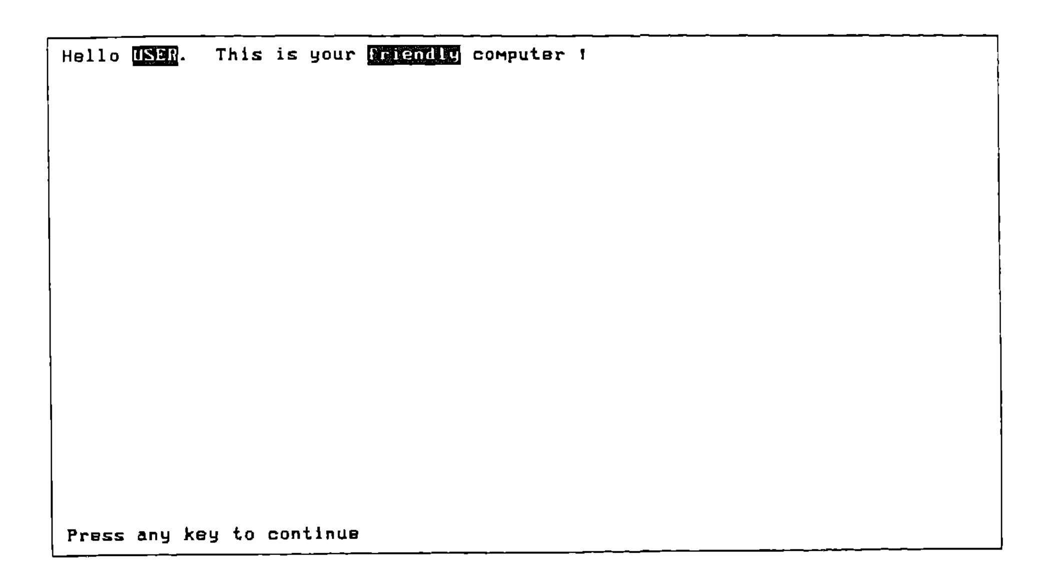
١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
OutStr$ = "Hello @USER|.  This is your @friendly| computer !"
SPos = 1: CLS : COLOR 15, 0
FOR Cnt = 1 TO LEN(OutStr$)
  IF MID$(OutStr$, SPos, 1) = "@" THEN COLOR 0, 15: SPos = SPos + 1
  IF MID$(OutStr$, SPos, 1) = "|" THEN COLOR 15, 0: SPos = SPos + 1
  PRINT MID$(OutStr$, SPos, 1); : SPos = SPos + 1
NEXT

────────────────────────── Immediate ──────────────────────────

Main: <Untitled>        Context: Program not running          00007:005
```

٢ – اضغط على Shift-F5 لتنفيذ البرنامج فترى ما يلى :



٣ – اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ – من قائمة File اختر Save واكتب PRINT.BAS كاسم للملف ثم اختر الشكل النصى لحفظ هذا البرنامج.

٥ – من قائمة File اختر New.

٦ – انتقل إلى الدرس الخامس عشر للاستمرار فى تسلسل التعلم.

الدرس المائة وعشرة

عبارة PRINT USING

الوصف

تستخدم عبارة PRINT USING فى تشكيل مخرجات سلاسل ومخرجات عددية بطريقة محددة، وتكوينها هو كما يلى :

```
PRINT USING format string; expression
```

جزء format string هو تعبير سلسلة توجد فيه رموز التشكيل المستخدمة فى طباعة البيانات. وجزء expression هو البيانات المراد طباعتها ويكون من النوع العددى أو نوع السلسلة. ويمكن طباعة أكثر من تعبير واحد باستخدام فواصل منقوطة لفصلها عن بعضها البعض. ورموز التشكيل لطباعة بيانات سلسلة تختلف عن رموز التشكيل لطباعة بيانات عددية. وفيما يلى وصف لرموز تشكيل بيانات السلسلة.

رمزالتشكيل	الفرضمنه
!	يتسبب فى طباعة أول رمز من تعبير السلسلة فقط.
\ \	يطبع n + 2 رمزاً من تعبير السلسلة حيث n هو عدد الفراغات الموجودة بين الشرطتين المائلتين للخلف. عندما تكون السلسلة أطول من n فتهمل الرموز الاضافية. أما إذا كانت السلسلة أقل من n فتضبط مخرجات السلسلة من ناحية اليسار مع ترك فراغات من ناحية اليمين.
&	ينسبب فى طباعة تعبير السلسلة كما هو.

وفيما يلى وصف لرموز التشكيل لطباعة بيانات عددية :

رمز التشكيل	الغرض منه
#	يستخدم هذا الرمز فى تحديد موقع الرقم. فإذا كان العدد له أرقام أقل عن المواصفة فيطبع العدد مرحلاً لليمين مع وجود فراغات سابقة للعدد. أما إذا كانت أرقام العدد أكبر من الوضع المحدد فتهمل الأرقام الأكثر.
.	يطبع علامة عشرية فى مكان ظهوره. وتقرب البيانات العددية عندما تكون هناك حاجة لذلك.
+	يسمح بطباعة اشارة العدد قبله أو بعده طبقاً لما هو محدد له.
-	يسمح بطباعة اشارة سالب.
**	يسمح بظهور نجوم فى الفراغات السابقة للعدد.
$$	يسمح بطباعة علامة دولار أمام القيمة العددية.
**$	يدمج تأثير النجوم وعلامات الدولار. وتملأ الفراغات السابقة للعدد بنجوم ثم تظهر علامة الدولار أمام العدد.
,	عندما تستخدم على يسار علامة عشرية فإنها تتسبب فى طباعة فاصلة كل ثلاث خانات من على يسار العلامة العشرية. أما إذا ما استخدمت فى نهاية سلسلة التشكيل فإنها تتسبب فى طباعة فاصلة.
^^^^	يحدد شكل الاس (E+xx). واستخدام خمس علامات منها يمكن من طباعة اعداد أكبر (E+xxx). ويتسبب استخدام العلامة العشرية فى ضبط الأرقام المعنوية من ناحية اليسار مع ضبط الأس.
_	هذه الشرطة التى تكون تحت حرف (فراغ فى هذه الحالة) تطبع الرمز التالى كثابت حرفى .
%	تتسبب فى طباعة اشارة النسبة المئوية عندما يكون العدد أكبر من الحقل. وعندما يقود التقريب إلى عدد يتعدى هذا الحقل فتطبع النسبة المئوية قبل العدد.

أقصى عدد للخانات التى يمكن تحديدها للأرقام هو 24. وعندما يزيد العدد عن ذلك تظهر رسالة خطأ بأن استدعاء الدالة غير صحيح.

التطبيقات

عبارة PRINT USING مريحة جداً عندما تكون هناك حاجة إلى التحكم فى تشكيل بيانات المخرجات. وعبارة PRINT USING بإمكانياتها القوية للتشكيل تكون اختياراً واضحاً عند طباعة التقارير. وفيما يلى أمثلة لهذه العبارة :

```
PRINT USING "!";"Dead":"On";"Arrival"
PRINT USING "/   /";"Dead";" On"; "Arrival"
PRINT USING "&";"Dead";" On":" Arrival"
PRINT USING "####";123456
PRINT USING "**###";123
PRINT USING "$$";123.00
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PRINT USING، ابدأ بتحميل بيسك السريع.

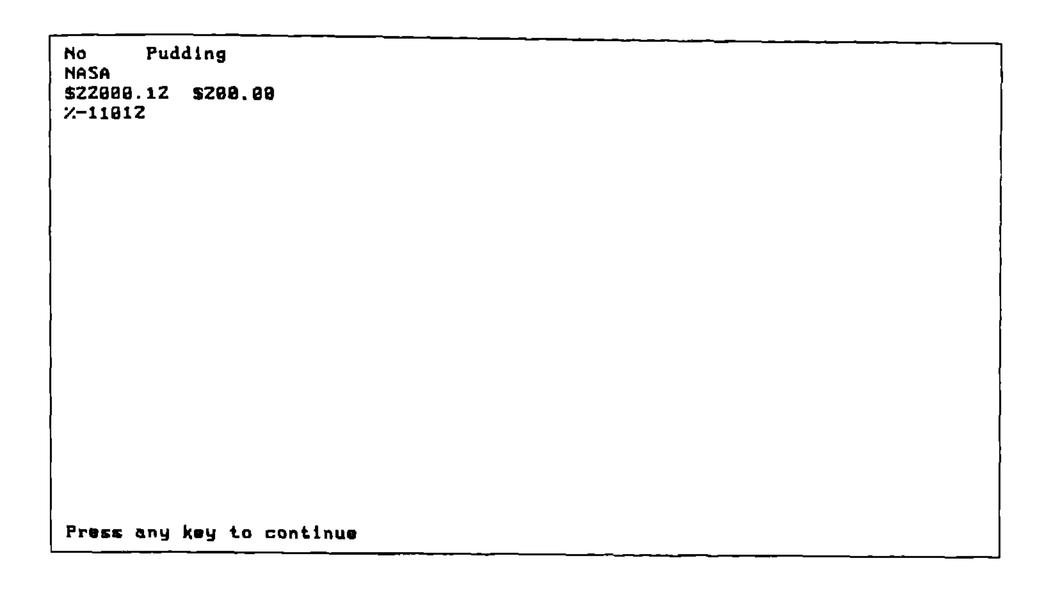
١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
'This program demonstrates the use of the PRINT USING statement.
CLS
DEFSTR A-F
Format1 = "\     \"
Format2 = "!"
Format3 = "$$####.##"
Format4 = "+####,##$"
PRINT USING Format1: "No": "Pudding"
PRINT USING Format2: "National"; "Aeronautic"; "Space"; "Administration"
PRINT USING Format3: 22000.12, 2001
PRINT USING Format4: 988 - 12000

                              Immediate

Main: <Untitled>      Context: Program not running              00011:033
```

٢ - نفذ البرنامج، لاحظ التشكيلات المختلفة لمخرجات البرنامج، لاحظ كذلك استخدام عبارة PRINT USING في تحقيق كل من هذه التشكيلات. اضغط على أي مفتاح للعودة إلى البرنامج.



```
No      Pudding
NASA
$22000.12  $200.00
%-11012

Press any key to continue
```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والسادس عشر للاستمرار في تسلسل التعلم.

الدرس المائة والحادى عشر

عبارتا # PRINT و PRINT # USING

الوصف

تتصرف عبارات # PRINT و PRINT # USING مثل عبارات PRINT و PRINT USING فيما عدى ان مقصد المخرجات يكون ملفا ، وتكوينها هو كما يلى :

تكوين عبارة # PRINT :

```
PRINT # filenum, expression list
```

جزء filenum هو رقم الملف المحدد بواسطة عبارة OPEN. جزء expression list هو قائمة بعناصر البيانات المراد كتابتها فى الملف . ارجع الى الدرس المائة وتسعة لمعرفة رموز التشكيل التى تستخدم فى هذه العبارة.

تكوين عبارة PRINT # USING :

```
PRINT # filenum, USING string expression, expression list
```

جزء filenum هو رقم الملف المحدد فى عبارة OPEN. وجزء USING يشبه جزء USING فى عبارة PRINT USING، وجزء string expression من جزء USING مكون بطريقة متماثلة. ويكون expression list عبارة عن قائمة عناصر البيانات المراد كتابتها بالتشكيل المقدم بواسطة جزء USING. ارجع الى الدرس المائة وعشرة لمعرفة مواصفات التشكيل المستخدمة فى جزء USING.

التطبيقات

عبارتا # PRINT و PRINT # USING ما هما الا جزء من ترسانة تشغيل الملفات القوية. واستخدامهما يقتصر على تمييز المبرمج ونوع التطبيق فقط.

عبارة # PRINT :

مثال١

```
OPEN "Session.Log" FOR OUTPUT AS #2
PRINT #2 "Session start date "; DATE$, "time ";TIME$
```

مثال٢

```
PRINT #3 Var1,Var2,Var3
```

مثال٣

```
PRINT #1 S1 S2 S3
```

عبارة PRINT # USING :

مثال١

```
OPEN "Issue.Lst" FOR RANDOM AS #10
FormatList$ = "\  \"
PRINT #10 USING FormatList$; "Plenipotentiary"
```

مثال٢

```
PRINT #1 USING "!"; "Save"; "Our"; "Souls"
```

عملية تقليدية

هذه العملية توضح استخدام عباراتى # PRIVT و PRINT # USING . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
'This program illustrates the use of the PRINT # and PRINT # USING
'statements. The program opens a file and writes data to it.  The data
'is provided by the program.
'The file is closed and then listed to demonstrate the effect of the
'PRINT # and PRINT # USING statements.

CLS
OPEN "Print.Fil" FOR OUTPUT AS #3
L1$ = "Earth Moving Equipment": L2$ = "12": L3$ = "$120,000"
'The PRINT # statement.
PRINT #3, L1$, L2$, L3$
L1$ = "Farm Equipment": L2$ = "22": L3$ = "$85,000"
PRINT #3, L1$, L2$, L3$
Format$ = "\   \"
PRINT #3, USING Format$; L1$, L2$, L3$
PRINT #3, USING "!"; "Save", "Our", "Souls"
CLOSE #3
```

Immediate

٢ – نفذ البرنامج، لاحظ استخدام عبارات # PRINT و PRINT # USING فى البرنامج.

٣ – لا يطبع البرنامج اى شىء على الشاشة. اختر Exit من قائمة File واكتب N للخروج من بيسك السريع. استخدم امر Type من نظام التشغيل DOS لترى محتويات الملف PRINT . FILE . وفيما يلى الملف الذى يكتب باستخدام امر Type من DOS .

```
C:\QB>type print.fil
Earth Moving Equipment      12           $120,000
Farm Equipment              22           $85,000
Farm 22   $85.0
SOS

C:\QB>
```

٤ – انتقل الى الدرس الثالث والستين للاستمرار فى تسلسل التعلم.

الدرس المائة والثانى عشر

عبارة PSET

الوصف

ترسم عبارة PSET نقطة على الشاشة عند احداثيات محددة. وتكوينها هو كما يلى :

```
PSET STEP (x,y),color
```

يحدد جزء STEP ان الاحداثيات نسبية للموقع الحالى وهو اختيارى. ويعطى جزء (x , y) احداثيات الشاشة للنقطة المراد رسمها . وجزء color يصف خاصية اللون للنقطة وهو اختيارى. وعندما يحذف هذا الجزء فيكون اللون المختار هو لون الامامية التقليدى.

التطبيقات

عبارة PSET مفيدة فى رسم نقاط على الشاشة دون الاعتماد على اشياء الرسومات الموجودة بالفعل على الشاشة. مثال ذلك يمكنك ان تستخدمها فى رسم نجوم فى السماء او فى اضافة مقطع للرسم . ويمكن ان تستخدم كذلك فى رسم رسومات خطوط. واستخدامها محدود بتطبيقات تكون مبرمجة فقط. وفيما يلى امثلة لعبارة PSET :

مثال١

```
FOR C = 1 TO 100
     PSET(C,100)
NEXT
```

مثال٢

```
FOR C = 1 TO 200
     PSET(100,INT(SIN(C)))
NEXT
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PSET فى رسم رسومات . استمر اذا كانت لديك امكانيات رسومات ملونة تدعم ذلك فقط. ابدا بتحميل بيسك السريع .

١ - اختر Open وحمل البرنامج PRESET. عدل عبارات PERSET الى ما هو مبين فى القائمة التالية :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          PRESET.BAS
'This program uses the PSET statement.
SCREEN 1:  COLOR 1, 3
CLS

FOR cnt = 1 TO 50
   PSET (65, cnt)

NEXT

VIEW (100, 10)-(150, 30), , 1
FOR cnt = 1 TO 50
   PSET (1, cnt)

NEXT

                          Immediate

Main: PRESET.BAS      Context: Program not running          C  00012:016
```

٢- نفذ البرنامج ، لاحظ استخدام عبارة PSET فى البرنامج.

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - ارجع الى الدرس الثامن والتسعين للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث عشر

عبارة RANDOMIZE

الوصف

نضع عبارة RANDOMIZE القيمة الابتدائية لمنتج الارقام العشوائية وتعرف عملية تحديد القيمة الابتدائية بأنها اعادة وضع قيمة للاساس reseeding. وتكوينها هو كما يلى :

```
RANDOMIZE numeric expression
```

اذا حذف التعبير العددى فيتوقف البرنامج ويسأل عن قيمة بالملقن :

```
Random number seed (-32768 to 32767) ?
```

وإلا فيستخدم التعبير العددى فى اعادة وضع قيمة الاساس لمنتج الارقام العشوائية .

التطبيقات

الغرض من استخدام عبارة RANDOMIZE هو منع دالة RAND من انتاج نفس تسلسل الارقام العشوائية . فاذا كان تسلسل الارقام العشوائية مختلفا فى كل مرة ينفذ فيها البرنامج فتستخدم دالة RANDOMIZE مع مؤشر مختلف فى كل تنفيذ وفيما يلى بعض الامثلة :

```
RANDOMIZE 22
RANDOMIZE 10 + LastVal%
RANDOMIZE TIMER
```

يستخدم آخر مثال القيمة التى تعود بواسطة دالة TIMER فى وضع قيمة ابتدائية لمنتج الارقام العشوائية .

عملية تقليدية

العملية التالية توضح استخدام دالة RANDOMIZE. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                                    F1=Help
                                  <Untitled>
'This program demonstrates the RANDOMIZE statement.
'This program generates random numbers using the RND function.
CLS
RANDOMIZE TIMER
PRINT "Numbers generated using the RND function:"
FOR Cnt = 1 TO 20
  PRINT INT((32768 - 0 + 1) * RND(Cnt \ 1.1) + 0),
NEXT

                                  Immediate

Main: <Untitled>        Context: Program not running                      00001:008
```

٢ - نفذ البرنامج . لاحظ مخرجات البرنامج واستخدام دالة RANDOMIZE فى البرنامج . لاحظ كذلك ان الارقام العشوائية اجبرت على ان تكون ارقاما صحيحة. فى هذا البرنامج توضع قيمة ابتدائية لمنتج الارقام العشوائية وعلى هذا فقد تختلف مخرجاتك عن المخرجات التالية :

```
Numbers generated using the RND function:
 15646          2653          13412         6399          31523
 31961          30557         8414          4399          28921
 21411          6995          5111          4841          21369
 29882          12236         884           21952         3851




Press any key to continue
```

٣ – اضغط على اى مفتاح للعودة الى البرنامج . من قائمة File اختر Save واحفظ هذا البرنامج كملف نص له الاسم RANDOM.BAS .

٤ – فى قائمة File اختر New مع اخلاء الشاشة.

٥ – انتقل الى الدرس المائة للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع عشر

عبارة READ

الوصف

تستخدم عبارة READ فى قراءة بيانات من عبارة DATA فى متغيرات وتكوينها هو كما يلى :

```
READ variable list
```

الجزء	الوصف
READ	كلمة من كلمات بيسك السريع المحجوزة.
variable list	قائمة متغيرات مفصولة عن بعضها البعض بفواصل وتقرأ فيها البيانات. وتتحدد القيم من عبارة DATA فى هذه المتغيرات.

ويحدث خطأ تحت الشروط التالية :

- اذا كانت هناك عناصر بيانات فى عبارة DATA اقل من المتغيرات الموجودة فى قائمة المتغيرات.

مثال

```
READ L1,L2,L3,L4,L5
..
..
DATA 100,24,234
```

- اذا لم يتفق نوع عنصر البيانات مع نوع المتغير الموجود فى عبارة DATA.

مثال

```
READ Name$,AcctNo
..
..
DATA 92855,Cliff Brooks
```

- اذا كان عنصر بيانات عددى اكبر مما يستطيع المتغير ان يحتويه.

مثال

```
READ  Month%
..
..
DATA 40000
```

يجب ان تقرأ عناصر متغيرات السجل عنصرا يتلو عنصر اخر فاذا كان عدد عناصر البيانات فى عبارة DATA يتعدى عدد المتغيرات الموجودة فى قائمة متغيرات عبارة READ فتقرأ عبارة READ التالية البيانات بدءا بأخر عنصر بيانات لم يقرأ .

مثال

```
READ Wun,Too,Thri
READ Fowr,Fif,Sicz
..
..
DATA 1,2,3,4,5,6
```

التطبيقات

عبارة READ مفيدة جدا فى تحميل البيانات فى متغيرات اثناء تنفيذ البرنامج. ودائما ماتستخدم عبارة READ مع عبارة DATA وعادة ما تستخدم مع عبارة RESTORE. وفيما يلى بعض الامثلة :

مثال١

```
READ FName$,MInit$,LName$,Address$
DATA David,W,Sullen,"1313 Mockingbird Ln, Metropolis"
```

لاحظ ان عنصر البيانات Address$ موضوع داخل علامتى تنصيص مزدوجة . وهذا لأن الفاصلة جزء من عنصر البيانات . يجب ان توضع البيانات من نوع السلسلة بين علامتى تنصيص مزدوجة اذا ما احتوت البيانات على فراغات سابقة او تابعة او اذا ما احتوت البيانات على نقطتين رأسيتين او فاصلة.

مثال٢

```
READ Qt%,Pnt%,Ltr%
READ Oz%,Lb%,Ton%
DATA 12,19,99
DATA 16,1000,1
```

مثال٣

```
DIM A(10) AS INTEGER
READ A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9),A(10)
DATA 10,9,8,7,6,5,4,3,2,1
```

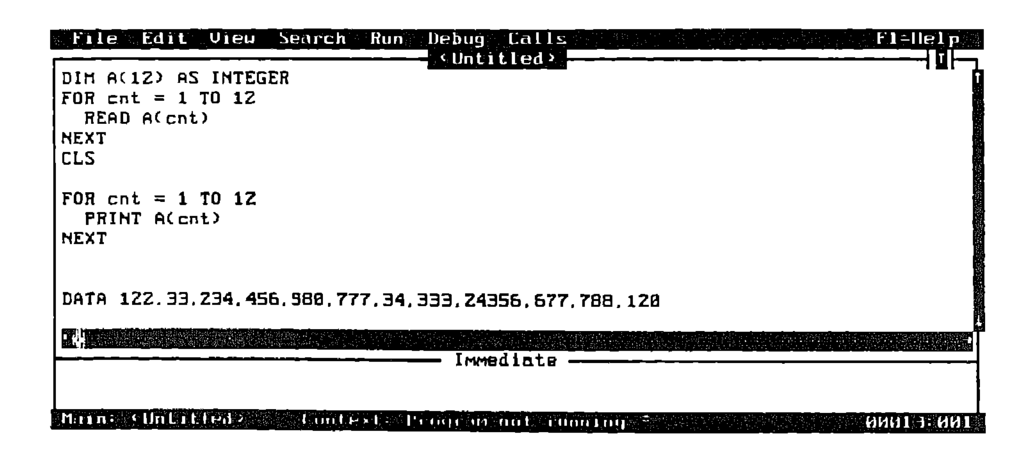
مثال٤

```
TYPE GeneralAccount
   Name AS STRING*30
   AcctType AS STRING*3
   AcctLimit AS DOUBLE
END TYPE
DIM NewAccount AS GeneralAccount
READ NewAccount.Name,NewAccount.AcctType,NewAccount.AcctLimit
DATA ACME Gag Gifts,NEW,1000.00
```

عملية تقليدية

هذه العملية تعطى مثالا لبرنامج يستخدم عبارة READ. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج ولاحظ استخدام عبارة READ فى تحميل بيانات داخل متغيرات.

```
122
33
234
456
980
777
34
333
24356
677
788
120






Press any key to continue
```

٣ - اضغط على اى مفتاح للعودة الى البرنامج. اضغط على Alt - F ثم اضغط على مفتاح الادخال وأكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس السابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس عشر

عبارة REDIM

الوصف

تستخدم عبارة REDIM في تعديل حجم الذاكرة المحدد لاى منظومة بعبارة REDIM ويجب ان تكون المنظومة ديناميكية $DYNAMIC وتكوينها هو كما يلى :

```
REDIM variable (subscripts) AS type
```

جزء variable هو اسم متغير صحيح فى بيسك السريع . وجزء (subscript) اختيارى ويستخدم فى تعريف ابعاد المنظومة. ويعرف جزء A Stype نوع المتغير. يمكن ان يكون النوع نوعا بسيطا (INTEGER او LONG او SINGLE او DOUBLE او STRING) او من النوع الذى يعرفه المستفيد.

جزء (subscripts) له التكوين التالى :

```
(lowerlimit TO upperlimit)
```

تتسبب عبارة REDIM فى اعتبار كل المنظومات على انها ديناميكية اثناء الترجمة. اثناء التنفيذ يعاد تحديد موقع البيانات الموجودة فى المنظومة قبل اعادة تحديد الابعاد وذلك بالحجم الجديد. وتفقد كل البيانات التى كانت فى المنظومة قبل اعادة تحديد الحجم. وتوضع قيمة صفر للمتغيرات العددية وقيمة فراغ لمتغيرات السلسلة.

يمكن استخدام عبارة REDIM فى تغيير حجم المنظومة الا انه لا يمكن استخدامها فى تغيير ابعاد المنظومة. ينتج المثال التالى رسالة بأن المنظومة لها أبعاد بالفعل .

```
REM $DYNAMIC
DIM TempArr(12,12)
..
RE! IM TempArr(12,12,12)
```

وينفذ المثال التالى بدون اخطاء خاصة بابعاد المنظومة.

```
REM $DYNAMIC
DIM TempArr(12,12)
..
REDIM TempArr(10,10)
```

التطبيقات

عبارة REDIM مفيدة فى التحكم فى متطلبات البرنامج من الذاكرة اثناء وقت التنفيذ حيث يمكن ان تتحدد المنظومات ويلغى تحديدها كلما كان ذلك مطلوبا. عندما يكون لبرنامج متطلبات ذاكرة كبيرة وليس لديه اتصال لمثل هذه الذاكرة فيمكن ان يتمتع البرنامج بميزة الذاكرة المتاحة عن طريق تحديد والغاء تحديد المنظومات اثناء وقت التنفيذ . وفيما يلى امثلة لعبارة REDIM :

مثال١

```
REDIM RT(34,34)
```

مثال٢

```
REM $DYNAMIC
DIM A1(1000,1000)
..
REDIM A1(100,100)
```

عملية تقليدية

هذه العملية توضح استخدام عبارة REDIM. ابدأ بتحميل بيسك السريع .

١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the REDIM statement. The program uses DATA
'statements to redimension an array and finds the smallest and
'largest number in the array each time.
Max = 15
DIM A(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A(Cnt)
  NEXT
  RETURN
```

```
FindMinMax:
  MinVal = A(1): MaxVal = A(1)
  FOR Cnt = 2 TO Max
     IF MinVal > A(Cnt) THEN
        MinVal = A(Cnt)
     END IF
     IF MaxVal < A(Cnt) THEN
        MaxVal = A(Cnt)
     END IF
  NEXT
  RETURN


Start:
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "First pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  READ Max
  REDIM A(Max)
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "Second pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,80,67,54,10
DATA 10
DATA 8,89,76,54,23,32,12,4,33,54

Immediate
Main: <Untitled>     Context: Program not running     00054:001
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة REDIM في البرنامج.

```
First pass
Minimum of array:  1        Maximum of array:  656
Second pass
Minimum of array:  4        Maximum of array:  89







Press any key to continue
```

٣ – ارجع الى البرنامج واحفظه على انه ملف نص مع اعطاء الاسم REDIM.BAS له مع اخلاء الشاشة.

٤ – انتقل الى الدرس السادس والتسعين للاستمرار فى تسلسل التعلم.

الدرس المائة والسادس عشر

عبارة REM

الوصف

تستخدم عبارة REM في اضافة تعليقات الى البرنامج . ويمكن تحقيق نفس التاثير من خلال استخدام الفاصلة . وتكوينها هو كما يلي :

```
REM remark
```

جزء remark هو سطر نصي يساعد في فهم وتتبع شفرة البرنامج . وعبارة ERM ليست عبارة للتنفيذ ولا تؤثر على تنفيذ البرنامج . ويمكن ان تظهر عبارات التنفيذ الاخرى على نفس السطر الواقعي لعبارة REM اذا ما استخدمت نقطتان رأسيتان كفاصل بينهما. كما تستخدم عبارات REM كذلك في احتواء اشباه الاوامر($STATIC او $DYNAMIC او $INCLUDE) في البرنامج وقد نوقشت هذه الاوامر في الدروس الخاصة بها.

التطبيقات

عبارة REM مريحة في تقديم ملاحظات توضيحية في البرنامج. كما انها ضرورية كذلك في تقديم اى واحد من اشباه الاوامر في البرنامج. وفيما يلي بعض الامثلة :

```
REM Variable declaration
REM Stand alone process for calculating Cyclical Redundancy Check
REM Output procedure for user-defined device USR:
'Note the variable Type2 is used as an alias for TypeCast
' also note that the loop variable is different
REM $INCLUDE: "Files.Bas"
'Simple interest calculation : Int! = 0.12 : Period% = 12
```

عملية تقليدية

هذه العملية توضح استخدام عبارة REM. ابدأ بتحميل بيسك السريع .

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
REM This program demonstrates the REM statement
REM The program illustrates the inaccuracy of computer addition.
'Notice the value of s after the 1000 iterations: it should be
'100 but is actually less than that.

CLS

FOR i = 1 TO 1000: s = s + .1: NEXT
PRINT s

                           Immediate

Main: <Untitled>     Context: Program not running          00009:008
```

٢ – نفذ البرنامج، لاحظ استخدام عبارة REM فى البرنامج واضغط على اى مفتاح للعودة الى البرنامج.

```
 99.99905






Press any key to continue
```

٣ – من قائمة File اختر New واكتب N لاخلاء الشاشة.

٤ – انتقل الى الدرس المائة والثامن عشر للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع عشر

عبارة RESET

الوصف

تغلق عبارة RESET كل الملفات المفتوحة على القرص، وتكوينها هو كما يلى :

```
RESET
```

ولا تستخدم اى قيم مع عبارة RESET للتمرير كما انها لاتعيد اى قيمة وتكتسب العبارة البيانات فى الذاكرة الاحتياطية النهائية للملفات وتغلق كل الملفات .

التطبيقات

عبارة RESET هى طريقة جيدة لضمان ان ملفات القرص تم تجديدها واغلاقها بطريقة مناسبة قبل فصل البرنامج، وتستخدم كأخر عبارة فى البرنامج قبل فصله . وفيما يلى مثال لها .

```
OPEN "Client.Act" FOR APPEND AS #1
..
PRINT #1, CLRec
..
RESET
END
```

عملية تقليدية

هذه العملية توضح عبارة RESET . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                              <Untitled>
'This program demonstrates the RESET statement. The program from
'Module 19 is modified to use the RESET statement instead of the CLOSE
'statement. A new file, PRINTZ.FIL, is created.

CLS
OPEN "PRINTZ.FIL" FOR OUTPUT AS #3
L1$ = "Earth Moving Equipment": L2$ = "12": L3$ = "$120,000"
'The PRINT # statement.
L1$ = "Farm Equipment": L2$ = "22": L3$ = "$85,000"
PRINT #3, L1$, L2$, L3$
Format$ = "\   \"
PRINT #3, USING Format$: L1$, L2$, L3$
PRINT #3, USING "!": "Save", "Our", "Souls"
RESET

                              Immediate

Main: <Untitled>        Context: Program not running                  00009:052
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة RESET في البرنامج .

٣ – ارجع الى البرنامج واختر New دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والتاسع والثلاثين للاستمرار في تسلسل التعلم.

الدرس المائة والثامن عشر

عبارة RESTORE

الوصف

تمكن عبارة RESTORE برنامج البيسك من اعادة قراءة بيانات من عبارات DATA عندما يتحدد رقم سطر او اسم سطر فتقرأ البيانات منه. وتكوينها هو كما يلى :

```
RESTORE line number or label
```

عندما ينفذ البرنامج عبارة RESTORE فتقرأ عبارة READ التالية البيانات من سطر محدد فإذا لم يكن هناك رقم او اسم سطر محدد فتقرأ البيانات من اول عبارة DATA موجودة فى البرنامج، وعندما يتحدد رقم او اسم للسطر فيجب ان توجد الاشارة له على مستوى الجزء الرئيسى . وتنقل كل عبارات DATA تلقائيا فى بيسك السريع الى شفرة على مستوى الجزء الرئيسى.

التطبيقات

تستخدم عبارة RESTORE فى قراءة نفس عبارة DATA اكثر من مرة واحدة. وهذا مفيد تحت شروط عديدة . تحميل منظومة هو احد هذه الشروط واعادة وضع قيم ابتدائية لمتغيرات لقيمها الاصلية هى شرط اخر. وفيما يلى بعض الامثلة :

```
..
READ q1, q2, q3
..
DATA 24.30.555
..
RESTORE
READ q1, q2, q3
```

يبين هذا المثال عبارات READ و DATA و RESTORE فى اعداد تقليدى فتقرأ أول عبارة READ القيم J q1 و q2 و q3. ويستمر البرنامج فى التنفيذ ليلتقى بعبارة RE-STORE وعند ذلك يسمح بإعادة قراءة قيم q1 و q2 و q3.

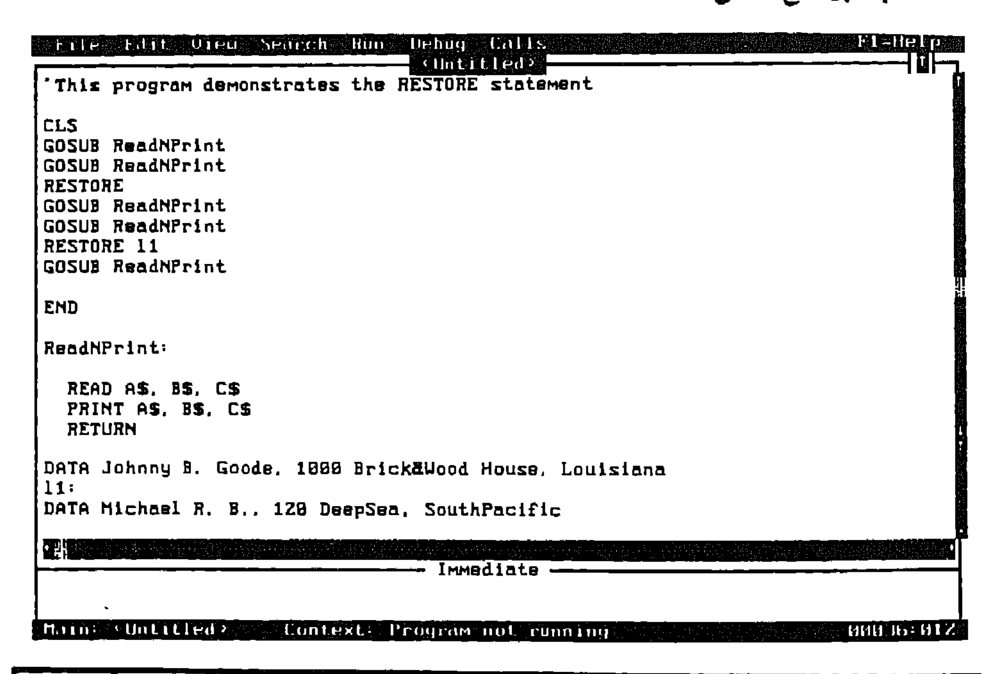
```
..READ St1$, St2$, St3$, t1%, t2%, R1!, R2!
..
DATA 1313, Mockingbird Ln., Munsters
Label1:
DATA 11,90
Label2:
DATA 0.33133, 9.78665
..
RESTORE Label1
READ t1%, t2%
..
RESTORE Label2
READ R1!, R2!
```

يوضح هذا المثال استخدام عبارة RESTORE مع اشارة اختيارية للاسم. لاحظ ان اول عبارة RESTORE تشير الى label1 والذى يسمح بالقراءة من عبارة DATA عند هذا الاسم . وتشير عبارة RESTORE الثانية الى label2 ويسمح ذلك بقراءة البيانات من عبارة DATA هذه .

عملية تقليدية

هذه العملية توضح استخدام عبارة RESTORE . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :



```
'This program demonstrates the RESTORE statement

CLS
GOSUB ReadNPrint
GOSUB ReadNPrint
RESTORE
GOSUB ReadNPrint
GOSUB ReadNPrint
RESTORE 11
GOSUB ReadNPrint

END

ReadNPrint:

  READ A$, B$, C$
  PRINT A$, B$, C$
  RETURN

DATA Johnny B. Goode, 1000 Brick&Wood House, Louisiana
11:
DATA Michael R. B., 120 DeepSea, SouthPacific
```

٢ - نفذ البرنامج. لاحظ المخرجات وكيفية استخدام البرنامج لعبارات RESTORE فى اعادة قراءة البيانات. اضغط على اى مفتاح للعودة الى البرنامج.

```
Johnny B. Goode               1000 Brick&Wood House       Louisiana
Michael R. B. 120 DeepSea     SouthPacific
Johnny B. Goode               1000 Brick&Wood House       Louisiana
Michael R. B. 120 DeepSea     SouthPacific
Michael R. B. 120 DeepSea     SouthPacific

Press any key to continue
```

٣ - من قائمة File اختر New واكتب N لا خلاء الشاشة.

٤ - انتقل الى الدرس السابع والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع عشر

عبارة RESUME

الوصف

تتسبب عبارة RESUME فى الاستمرار فى تنفيذ البرنامج بعد اصطياد احد الاخطاء .
وتكوينها هو كما يلى حيث إنها تأخذ ثلاثة اشكال :

التكوين الاول :

```
RESUME 0
```

يجعل هذا التكوين البرنامج يستمر عند السطر الذى حدث عنده الخطأ وجزء 0 من التكوين يمكن اهماله بدون ان يتغير التأثير.

التكوين الثانى :

```
RESUME NEXT
```

يجعل هذا التكوين البرنامج يستمر عند السطر الذى يلى السطر الذى حدث فيه الخطأ مباشرة.

التكوين الثالث:

```
RESUME line number|line label
```

يجعل هذا التكوين البرنامج يستمر عند رقم سطر او اسم سطر معين، ورقم السطر او اسم السطر الذى يشار اليه يجب ان يكون على مستوى الجزء، ومن الافضل تجنب هذا التكوين بحيث يمكن استمرار التنفيذ بغض النظر عن موقع حدوث الخطأ .

عندما تستخدم عبارة RESUME خارج جزء معالجة الخطأ فتنتج رسالة بأن عبارة RE-SUME بدون خطأ . وعندما يحدث خطأ داخل تكوين DEF FN فتستمر عبارة RESUME وعبارة RESUME NEXT فى تنفيذ البرنامج عند السطر الذى يحتوى على الدالة.

التطبيقات

تستخدم عبارة RESUME بالاتصال مع اجزاء اصطياد الخطأ ومعالجة الخطأ مثل عبارة ON ERROR GOTO. وفيما يلى بعض الامثلة :

مثال١

```
ON ERROR GOTO 1000
..
1000
     PRINT "Error in line "; ERL
     RESUME NEXT
```

مثال٢

```
ON ERROR GOTO ErrHandler
..
ErrHandler:
     ..
     RESUME
```

عملية تقليدية

هذه العملية توضح عبارة RESUME . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                 <Untitled>
'This program demonstrates the RESUME statement. The program asks
'for a filename to list, and ON ERROR gives a message and continues to
'ask for a filename.
ON ERROR GOTO EH
CLS
Begin:
INPUT "Enter filename to list: "; FileName$
OPEN FileName$ FOR INPUT AS #1

DO WHILE NOT EOF(1)
   LINE INPUT #1, Line$
   PRINT Line$
LOOP

EH:
  PRINT "Error number: "; ERR
  RESUME Begin

─────────────────────────── Immediate ───────────────────────────

Main: <Untitled>      Context: Program not running               00017:015
```

٢ – نفذ البرنامج . لاحظ استخدام عبارة RESUME في البرنامج.

```
Enter filename to list: ? Music
Error number:  53
Enter filename to list: ? NoName
Error number:  53
Enter filename to list: ?
```

٣ – ارجع الى البرنامج واختر New دون ان تحفظ هذا البرنامج .

٤ – انتقل الى الدرس المائة والسابع عشر للاستمرار في تسلسل التعلم.

الدرس المائة والعشرون

دالة RIGHT$

الوصف

تعيد دالة RIGHT$ عدد الرموز المحدد على اقصى يمين تعبير سلسلة وتكوينها هو كمايلى:

```
RIGHT$(string expression,num)
```

الجــــزء	الوصـــــف
RIGHT$	كلمة من كلمات بيسك المحجوزة.
string expression	سلسلة مـؤشـر والتى يعـود منهـا عـدد num الرمـوز الموجـودة على أقـصى اليمين. ويمكن أن يكون تعبـيـر سلسلة أو متغير أو ثابت.
num	عدد الرموز الموجودة فى أقصى اليمين والتى تعود من تعبير السلسلة. فإذا كان num مساوياً لطول تعبير السلسلة فتعيد الدالة RIGHT$ تعبير السلسلة كله.

التطبيقات

دالة RIGHT$ هى احدى وسائل تشغيل السلاسل فى بيسك السريع واستخداماتها محدودة بالتخيلات والتطبيقات فقط. وفيما يلى بعض الامثلة.

مثال١

```
FullName$ = "George B. Shaw"
PRINT RIGHT$(FullName$,4)
```

مثال٢

```
Path$ = "C:\BASIC\QB4"
FileName$ = RIGHT$(Path$,3) + "\NewComp.Bas"
PRINT FileName$
```

مثال٣

```
FullName$ = "George B. Shaw"
PRINT RIGHT$(FullName$,40)
```

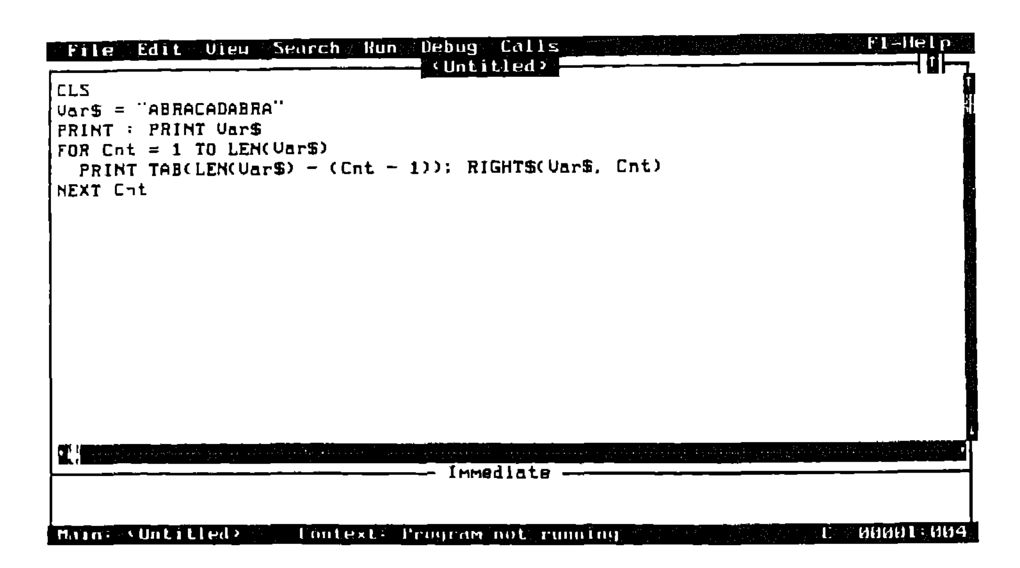
مثال٤

```
FullName$ = "Mark Meslone"
PRINT RIGHT$(FullName$,0)
```

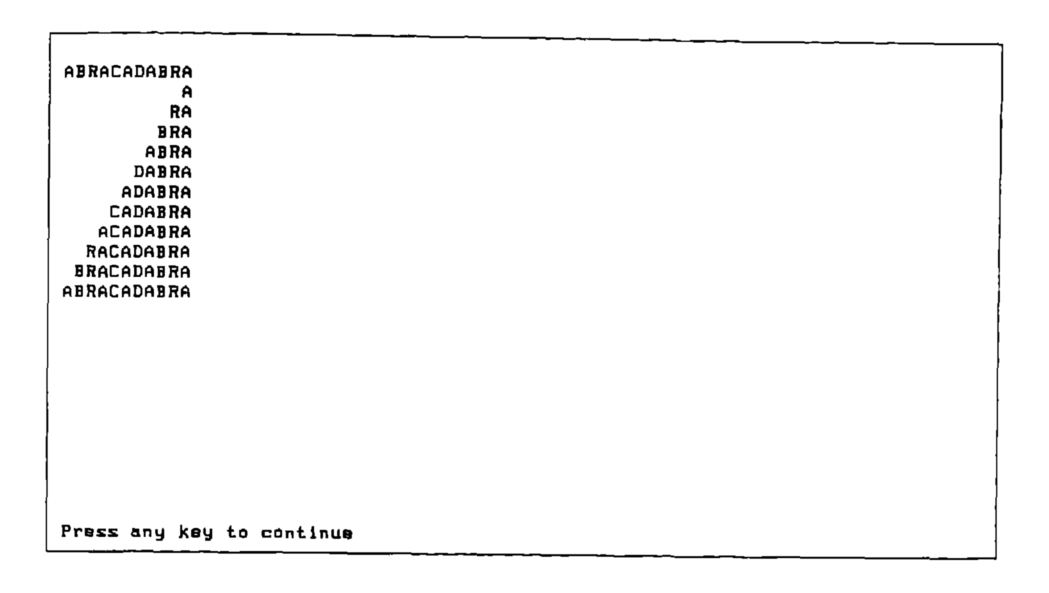
عملية تقليدية

هذه العملية توضح برنامجا بسيطا يستخدم دالة RIGHT$. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



٢ – نفذ البرنامج ولاحظ استخدام دالة RIGHT$ فى البرنامج.



٣ – اضغط على اى مفتاح للعودة الى البرنامج، اضغط على Alt - F ثم اضغط على مفتاح الادخال وأكتب N لإخلاء الشاشة.

٤ – انتقل الى الدرس الثالث والسبعين للاستمرار فى تسلسل التعلم.

الدرس المائة والحادى والعشرون

دالة RND

الوصف

تعيد دالة RND رقما عشوائيا يقع بين 0 و 1. والقيمة التى تعود تكون من النوع فردى الدقة. وتكوينها هو كما يلى :

```
RND (n)
```

القيمة التى تعيدها دالة RND تتحدد بواسطة n والتى تكون تعبيرا عدديا . ويصف الجدول التالى العلاقة :

N	النتيجـــة
n < 0	بعض الأرقام العشوائية بغض النظر عن قيمة n.
n > 0	الرقم العشوائى التالى فى التسلسل.
أو محذوفه	
n = 0	اخر رقم عشوائى تم انتاجه.

فإذا لم توضع قيمة ابتدائية لمنتج الارقام العشوائية باستخدام دالة RANDOMIZE فيتم انتاج نفس الارقام حتى اذا ما كان التعبير العددى اكبر من 0.

التطبيقات

دالة RND هى دالة مريحة عندما تكون هناك حاجة الى انتاج سلسلة من الارقام العشوائية اثناء تنفيذ البرنامج . كمثال لمثل هذا الموقف عملية رسم النجوم فى إحدى المباريات المرئية على الشاشة. وفيما يلى بعض الامثلة.

```
T! = RND(3)
NLoc% = INT((32-10+1)*RND+10)
```

يستنتج هذا المثال رقما عشوائيا صحيحا يقع بين 32 و 10 وذلك من قيمة لها دقة فردية تعيدها دالة RND . وهذه العملية توضح بصورة افضل فى المثال التالى .

```
Integer Random Number = INT ((Upper limit - Lower limit + 1)
                        * RND + Lower limit)
```

عملية تقليدية

هذه العملية توضح استخدام دالة RND. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                        <Untitled>
'This program generates random numbers using the RND function.
CLS
PRINT "Numbers generated using the RND function:"
FOR Cnt = 1 TO 20
  PRINT RND(Cnt \ 1.1),
NEXT

                         Immediate

Main: <Untitled>      Context: Program not running          00005:018
```

٢ - نفذ البرنامج ولاحظ استخدام دالة RND فى البرنامج. لاحظ ان المؤثر \ يستخدم بدلا من/ فى قسمة Cnt للحصول على الجزء الكسرى. وحيث إننا لم نضع قيما ابتدائية لمنتج الارقام العشوائية بعبارة RND فيجب ان ينتج البرنامج نفس تسلسل الارقام العشوائية . حاول ان تنفذ البرنامج مرة اخرى وتأكد مما اذا كان ذلك صحيحا ام لا .

```
Numbers generated using the RND function:
 .7107346      .99058       .8523988      .3503776      4.363585E-02
 8.977669E-02               .5111076      .7553547      .935394
 .3444337      .2073416     .5360375      .2351913      .7695047
 .2087399      .9520085     .9767802      .8262324      .4440745
 .2794519

Press any key to continue
```

٣ – اضغط على اى مفتاح للعودة الى البرنامج . من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ – انتقل الى الدرس المائة والثالث عشر للاستمرار فى تسلسل التعلم.

الدرس المائة والثاني والعشرون

عبارة RUN

الوصف

تستخدم عبارة RUN في تنفيذ برامج اخرى من داخل احد برامج بيسك السريع وتكوينها هو كما يلي :

```
RUN line|file spec
```

القضيب الرأسي (|) الموجود بين line و filespec يحدد أن أحد هذين المؤشرين فقط هو المستخدم . جزء line هو رقم السطر الذي يبدأ عنده تنفيذ البرنامج . وجزء filespec هو اسم ملف البرنامج المراد تنفيذه . وينفذ البرنامج كما لو كان يبدأ التنفيذ من البداية : كل الملفات مغلقة وقيم المتغيرات الابتدائية يعاد وضعها. ولا يمكنك ان تمرر معلومات الى برامج اخرى او اقتسام متغيرات.

التطبيقات

تستخدم عبارة RUN في تنفيذ برامج اخرى من برنامج ينفذ حاليا وفي إعادة بدء تنفيذ البرنامج الحالي من عند رقم سطر معين. وفيما يلي امثلة لعبارة RUN.

مثال١

```
PRINT "Please wait .."
RUN "Screen"
```

مثال٢

```
10  Wrt = 23
20  PRINT FNDblWrt(Wrt)
30  Wrt = Wrt * 2
40  PRINT FNDblWrt(Wrt)
50  RUN 30
```

في هذا المثال تكون قيمة Wrt مساوية 0 وتطبع عبارة PRINT هذه القيمة لأن التحديد السابق لـ Wrt لا يكون له اي تاثير.

عملية تقليدية

هذه العملية توضح استخدام وتاثير عبارة RUN كما هى محددة فى المثال التالى فى قسم التطبيقات السابق. ابدا بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                        F1=Help
                              <Untitled>
'This program demonstrates the RUN statement.
DEF FnDbl (i AS INTEGER) = i * 2

CLS
Wrt = 3
PRINT FnDbl(Wrt), Wrt
10 Wrt = Wrt * 2
PRINT FnDbl(Wrt), Wrt
IF Wrt = 1 THEN END
IF Wrt >0 THEN RUN 10
                              Immediate
Main: <Untitled>      Context: Program not running              00010:022
```

٢ – نفذ البرنامج. لاحظ المخرجات التالية وتاثير عبارة RUN على قيم المتغير.

```
 6              3
 12             6
 0              0



Press any key to continue
```

٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظه.

٤ – انتقل الى الدرس المائة والرابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والعشرون

دالة SADD

الوصف

تعطى دالة SADD عنوان تعبير سلسلة محدد. وتكوينها هو كما يلى :

```
SADD(string exp)
```

جزء string exp يكون متغير سلسلة او عنصراً من عناصر منظومة سلاسل. وغير مسموح بسلاسل ثابتة الطول. والعنوان الذى يعود هو فرع من قطاع البيانات الحالى. ولا يعمل استخدام SADD فى قطاعات غير قطاع بيانات بيسك السريع.

التطبيقات

تستخدم دالة SADD فى معظم الاحيان فى برمجة بخليط من اللغات حيث تكتب الاجزاء بلغات غير لغة بيسك السريع. وحيث انه يمكن ان تتحرك السلسلة داخل الذاكرة اثناء تنفيذ البرنامج فيجب ان تستخدم دالة SADD بحذر. اضافة رموز الى تعبير سلسلة المؤشر يقود الى حدوث خطأ وقت التنفيذ . وفيما يلى مثال لدالة SADD.

```
T$ = "ABCDEFGH"
PRINT SADD(T$)
```

عملية تقليدية

حيث إننا لا نستطيع ان نفترض ان القارىء لديه امكانية اتصال بمترجم لغة C من ميكروسوفت أو انه لديه معرفة بالبرمجة بلغة C فإن المثال الموجود فى هذا القسم محدود بتقديمه توضيحا بسيطا لنوع القيمة التى تعود بواسطة دالة SADD . ابدا بتحميل بيسك السريع .

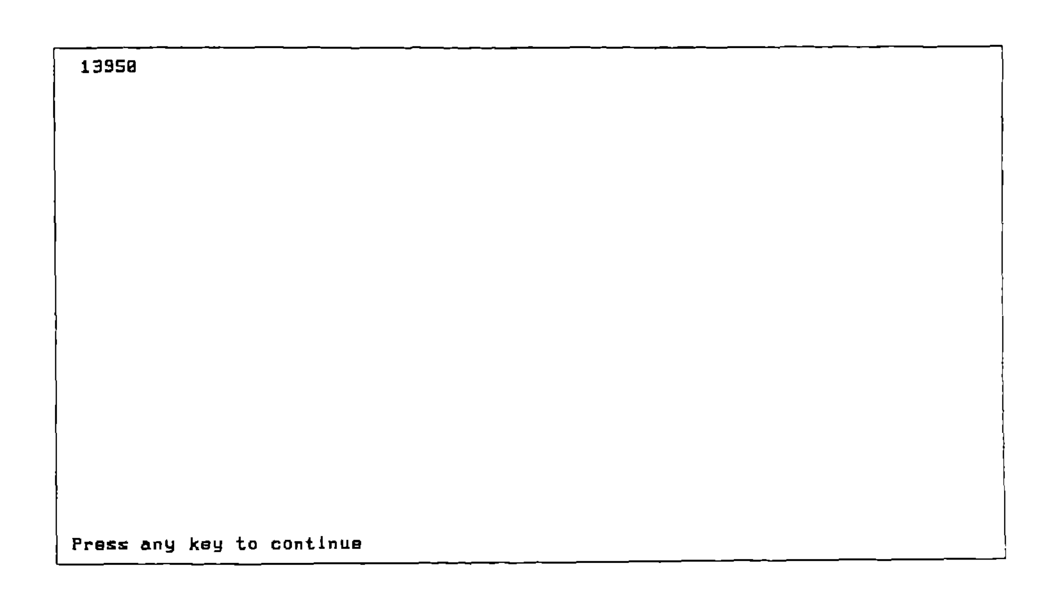
١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
'This program demonstrates the SADD function.
CLS
T$ = "Example string"
PRINT SADD(T$)

                              Immediate

Main: <Untitled>      Context: Program not running          00004:015
```

٢ – نفذ البرنامج. تكون المخرجات على النحو التالى :



٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والسادس والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع والعشرون

دالة وعبارة SCREEN

الوصف

تستخدم الكلمة المحجوزة SCREEN كدالة او كعبارة . وعندما تستخدم كدالة فإنها تعيد رمزا عند احداثيات الشاشة المعطاة. وعندما تستخدم كعبارة فإنها تعرف كيفية اعداد الكمبيوتر. وكل من الاستخدامين مذكور فى هذا الدرس .

دالة SCREEN: تكوينها هو كما يلى :

```
SCREEN (row, col, color flag)
```

جزءاً row و col هما احداثيات الصف والعمود وجزء color flag هو مؤشر اختيارى. تعيد SCREEN اللون عند الاحداثيات المستخدم عندها color flag وتعيد رمز قيمة ASCII عند الاحداثيات غير المستخدم عندها color flag . وتقوم كل المؤشرات بقيم عددية صحيحة وتكون الاقواس ضرورية حول المؤشرات.

عبارة SCREEN: تحدد عبارة SCREEN الخواص الخاصة بعرض الكمبيوتر. وتستخدم العبارة فى اختيار حالة العرض المناسبة لبرنامج تطبيق معين ونظام كمبيوتر معين. وتكوينها هو كما يلى :

```
SCREEN mode, color switch, active page, visual page
```

جزء mode عبارة عن تعبير عددى صحيح يصف حالة الشاشة . والجدول التالى يوضح القيم المختلفة للحالات وتأثيراتها المصاحبة لها ومتطلباتها.

الحالة	التاثيرات والمتطلبات
0	هذه هى الحالة النصية المعتادة (25 x 40 او 25 x 80 او 43 x 40 او 50 x 40 او 43 x 80 او 50 x 80) وحجم الرمز 8x8 (او 14 x 8 او 14 x 9 او 9x16 نقطة رسم مع بطاقة EGA او بطاقة VGA). وتسمح حتى 16 لونا و 2 خاصية او 16 لونا و 16 خاصية مع بطاقة EGA.
1	رسومات متوسطة الثبات (200 x 320 نقطة رسم) وعرض نصوص40x25 و16 لوناً

الحالة	التاثيرات والمتطلبات
	و 4 خواص مع بطاقة EGA وتدعم بطاقات CGA و EGA و VGA و MCGA.
2	رسومات مرتفعة الثبات (200 x 640) وعرض نصوص 25 x 80 و 16 لوناً و 2 خاصية مع بطاقة EGA او بطاقة VGA. وتدعم بطاقات CGA و EGA و VGA و MCGA.
7	رسومات متوسطة الثبات (200 x 320) وعرض نصوص 25 x 40 وصفحات متعددة الشاشات و 16 لوناً مع 16 خاصية وتتطلب بطاقة EGA او بطاقة VGA.
8	رسومات مرتفعة الثبات (200 x 640) وعرض نصوص 25 x 80 وصفحات متعددة الشاشات و 16 لوناً مع 16 خاصية وتتطلب بطاقة EGA او بطاقة VGA.
9	رسومات معززة (300 x 640) وعرض نصوص 25 x 80 او 43 x 80 برموز 8 x8 او 14 x 8 و 64 لوناً مع 16 خاصية او 16 لوناً مع 4 خواص طبقا لما اذا كانت الذاكرة بها بطاقة EGA او VGA وصفحات متعددة الشاشات وتتطلب EGA او VGA.
10	مثل الحالة 9 وتسمح بعدد 9 الوان شبيهة مع 4 خواص.
11	رسومات مرتفعة الثبات جدا (480 x 640) وعرض نصوص 30 x 80 او80x60 برموز 8x8 او 8x16 و 256K لوناً و 2 خاصية وتتطلب EGA او VGA.
12	مثل الحالة 11 وتسمح ب 256K لوناً من 16 خاصية وتتطلب VGA.
13	رسومات متوسطة الثبات (320x200) وعرض نصوص 40x25 و 256K لوناً و256 خاصية وتتطلب VGA او MCGA.

جزء color switch فى تكوين عبارة SCREEN يحدد ما اذا كان اللون معروضا ام لا على شاشات مركبة. و color switch هو قيمة تقع فى المدى من 0 الى255 وعندما لا تكون القيمة صفراً فتعرض صوراً ابيض واسود فقط ويعرض اللون عندما تكون صفراً. وفى حالة الشاشة 0 فتتحول القيمة، وتهمل القيمة فى حالات الشاشة من 2 واكثر.

جزء active page هو صفحة الشاشة التى تكتب فيها عبارات الرسومات. وجزء visual page هو الجزء المعروض حاليا.

وعند استخدام حالة الشاشة 0 مع عرض IBM احادى اللون وضابط طابع فيكون التاثير كما يلى :

Mode	0
RowsxCol	25x80
Attributes	16
Colors	3
Resolution	720x350
Pages	1

ومع ضابط الرسومات الملونة CGA من طرازIBM فما يلى هو حالات الشاشة وتاثيراتها.

Mode	*RowsxCols*	*Colors*	*Resolution*	*Pages*
0	40x25,80x25	16,16	320x200, 640x200	8 4
1	40x25	4	320x200	1
2	80x25	2	640x200	1

ومع ضابط رسومات معززة EGA تكون حالات الشاشة وتاثيراتها كما يلى :

Mode	*RowsxCols*	*Display*	*Attr.*	*Colors*	*Res.*	*Page, Page size*
0	40x25	C	16	16	320x200	8,NA
	40x25	E	''	64	320x350	''
	40x43	''	''	''	''	''
	80x25	C	''	16	640x200	''
	80x25	E	''	64	640x350	''
	80x25	C	''	16	640x200	''
	80x25	M	''	3	720x350	''
	80x43	E	''	64	640x350	4,NA
	80x43	M	16	3	720x350	''
1	40x25	NA	4	16	320x200	1,16K
2	80x25	''	2	''	640x200	''
7	40x25	''	16	''	320x200	1,32K
8	80x25	''	''	''	640x200	1,64K
9	80x25	E	4	64	640x350	''
	80x43	''	''	''	''	''
	80x25	''	16	''	''	1,128K
	80x43	''	''	''	''	''
10	80x25	M	4	9	''	1,64K
	80x43	''	''	''	''	''

ومع منظومة رسومات مرئية VGA تكون حالات الشاشة وتاثيراتها كما يلى :

Mode	RowsxCols	Attr.	Colors	Res.	Page, Page size
0	40x25	16	64	360x400	8,NA
	40x43	''	''	320x350	''
	40x50	''	''	320x400	4,NA
	80x25	''	''	720x400	8,NA
	80x43	''	''	640x350	4,NA
	80x43	''	3	720x350	''
	80x50	''	64	640x400	''
	80x50	''	3	720x400	''
1	40x25	4	16	320x200	1,16K
2	80x25	2	''	640x200	''
7	40x25	16	''	320x200	1,32K
8	80x25	''	''	640x200	1,64K
9	80x25	''	64	640x350	1,128K
	80x43	''	''	''	''
10	80x25	4	9	''	1,64K
	80x43	''	''	''	''
11	80x30	2	256K	640x480	''
	80x60	''	''	''	''
12	80x30	16	''	''	1,256K
	80x60	''	''	''	''
13	40x25	256	''	320x200	1,64K

ومع منظومة رسومات متعددة الالوان MCGA تكون حالات الشاشة وتاثيراتها كما يلى :

Mode	RowsxCols	Attr.	Colors	Res.	Page, Page size
0	40x25	16	NA	320x400	8,NA
	80x25	''	''	640x400	''
1	40x25	4	''	320x200	1,16K
2	80x25	2	''	640x200	''
11	80x30	''	256K	640x480	1,64K
	80x60	''	''	''	''
13	40x25	256	''	320x200	''

التطبيقات

تستخدم عبارة SCREEN فى تحديد حالة العرض المستخدمة فى برنامج تطبيق معين. ويجب ان يتم اختيار الحالة بعناية ويعتمد ذلك على نظم المكونات المتاحة . واختيار الحالة الصحيحة للشاشة للعمل فيها يكون مهما عندما يكون البرنامج متداخلا جدا او ينتج ويستخدم نوافذ او يستخدم رسومات. ويجب اختيار الحالة المناسبة للشاشة عند استخدام عبارات رسومات مثل DRAW و CIRCLE. وفيما يلى بعض الامثلة لعبارة SCREEN:

مثال١

```
SCREEN 2
..
```

مثال٢

```
SCREEN 1: COLOR 1
LINE (0,0)-(319-199)
..
```

مثال٣

```
SCREEN 2
DRAW D$
```

عملية تقليدية

توجد امثلة لعبارة SCREEN فى الدروس 76 و 35 و 16 و 97. وهذه العملية توضح دالة SCREEN فى صورة مبسطة. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                           <Untitled>
'This program illustrates the use of the SCREEN function.
SCREEN 0
CLS
FOR Cnt = 65 TO 75
    PRINT CHR$(Cnt);
NEXT
PRINT
PRINT "Results of the SCREEN function"

FOR Cnt = 1 TO 10
   V = SCREEN(1, Cnt)
   PRINT V
NEXT

                           Immediate

Main: <Untitled>      Context: Program not running        00001:001
```

٢ - نفذ البرنامج. لاحظ استخدام دالة SCREEN وعبارة SCREEN فى البرنامج. وتشبه المخرجات ما يلى :

```
ABCDEFGHIJK
Results of the SCREEN function
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74

Press any key to continue
```

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس السادس والسبعين للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس والعشرون

عبارة دالة و SEEK

الوصف

تستخدم SEEK كدالة وكعبارة . وتضع عبارة SEEK مشير الملف عند الموقع المحدد. وتعطى دالة SEEK موقع مشير الملف داخل الملف. وتهمل عبارات ودوال SEEK مع الوحدات التالية : SCRN و CONS و : KYBRD و COMx و LPTx حيث x رقم صحيح.

عبارة SEEK : وتكوينها هو كما يلى :

```
SEEK #filenum, pos
```

جزء filenum هو رقم الملف المحدد فى عبارة OPEN . جزء pos هو الموقع الذى ينقل اليه مشير الملف. وهذا هو المكان الذى تبدأ فيه عملية القراءة او عملية الكتابة التالية. ويمكن ان تصل قيمة pos حتى 2,147,483,647 كحد اقصى. ومع ملفات الاتصال العشوائى تكون pos هى رقم السجل . ومع الملفات المفتوحة بأنها ملفات BINARY او INPUT و OUTPUT او APPEND يكون pos هو موقع البايت فى الملف ودائما ما يكون pos اكبر من صفر.

دالة SEEK : وتكوينها هو كما يلى :

```
SEEK(filenum)
```

جزء filenum هو نفسه مثل ما هو موجود فى عبارة SEEK . والقيمة التى تعود من دالة SEEK تقع بين 1 و 647 ,483 ,147 ,2. وعند استخدامها مع ملفات اتصال عشوائى فإن القيمة التى تعود هى موقع السجل التالى . ومع الملفات المفتوحة على انها BINARY او IN- PUT او OUTPUT او APPEND فإن القيمة التى تعود تكون موقع البايت التالى .

التطبيقات

تستخدم عبارة SEEK فى القفز هنا وهناك داخل الملف اثناء تشغيله. وعبارات SEEK المحسوبة هى ميزة للاتصال بالسجلات الموجودة فى الملف. ودالة SEEK لها نفس العمل فى الاساس مثل دالة LOC. وفيما يلى بعض الامثلة :

```
OPEN "Inven.Dat" FOR BINARY AS #1
..
SEEK #1,256
OPEN "Address.Dat" FOR RANDOM AS #1 LEN = 200
..
SEEK #1,10
```

فى المثال السابق تنقل عبارة SEEK المشير الى السجل العاشر. وفى المثال الذى يسبقه تنقل عبارة SEEK المشير الى موضع البايت 256 . وفيما يلى امثلة لدالة SEEK :

```
SEEK #3, SEEK(3) - (LEN(RecordLen) * 3)
```

المثال السابق يستخدم دالة SEEK فى عمل عبارة SEEK محسوبة. ينتقل مشير السجل الى الخلف بثلاثة سجلات.

عملية تقليدية

هذه العملية توضح عبارة SEEK . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the SEEK statement. The program uses SEEK
'to read in a particular record from the file created in Module 74.

TYPE CustType
   CustName AS STRING * 25
   CustNum AS INTEGER
   CustType AS STRING * 2
END TYPE

DIM Customer AS CustType

OPEN "Cust.Fil" FOR RANDOM AS #2
SEEK #2, 2
GET #2, , Customer
PRINT LOC(2); Customer.CustName, Customer.CustNum, Customer.CustType

CLOSE #2

                         Immediate

Main: <Untitled>      Context: Program not running            00017:009
```

٢ – نفذ البرنامج ، لاحظ استخدام عبارة SEEK في البرنامج.

```
 2 More Money Corp.                          1ZZZ         C




















Press any key to continue
```

٣ – اضغط على اى مفتاح للعودة الى البرنامج.

٤ – اختر New من قائمة File واختر عدم حفظ هذا البرنامج.

٥ – انتقل الى الدرس الحادى والثمانين للاستمرار فى تسلسل التعلم.

الدرس المائة وستة وعشرون

عبارة SELECT CASE

الوصف

عبارة SELECT CASE هى طريقة اكثر ترتيبا فى كتابة عبارة IF THEN ELSE متعددة المستويات. وتكوين عبارة SELECT CASE هو كما يلى :

```
SELECT CASE test exp
   CASE test1
     statements
   CASE test2
     statements
  ..
   CASE ELSE
     statements
END SELECT
```

وفيما يلى وصف اجزاء التكوين:

الجزء	الوصف
test exp	هذا هو التعبير الذى يتم تقويمه لتحديد اى فرع تتفرع اليه عبارة SELECT CASE لتنفيذه .
test1, test 2	هذه هى النتائج الممكنة لتعبير الاختبار test exp .
statements	عبارات بيسك السريع التى تنفذ عندما تكون نتيجة test exp صحيحة بالنسبة الى test 1 او test 2 .

test1 و test2 لهما الصيغ التالية :

```
CASE test1, test2,..
CASE test1 TO test2
CASE test1 IS (<,>,<=,>=,<>,=)
```

اول صيغة هى قائمة بالنتائج الممكنة لتقويم التعبير المختبر. وتنفذ مجموعة العبارة عندما تكون النتيجة الفعلية واحدة من القائمة . وتعطى الصيغة الثانية مدى لقيم من test1 الى test2 وتنفذ مجموعة العبارة عندما تقع النتيجة داخل المدى . والصيغة الثالثة علاقية فى طبيعتها . ويمكن استخدام اى من المؤثرات العلاقية الموجودة بين الاقواس والاقواس نفسها غير مشمولة. وتنفذ مجموعة العبارة هذه عندما تقوم العملية العلاقية بأن قيمتها صحيحة TEUE .

ويقوم جزء CASE ELSE فى التكوين للتعامل مع النتائج الممكنة الاخرى غير الموجودة فى عبارات CASE السابقة . وتنفذ مجموعة العبارة هذه عندما لا يتحقق اى من الاختبارات التالية اى عندما تكون نتيجتها كلها FALSE . وجزء CASE ELSE اختيارى .

وتكوين SELECT CASE مثير عندما يتحقق شرط واحد وتنفذ مجموعة العبارة. وهذا هو سبب ان عبارة SELECT CASE عبارة عن طريقة اكثر ترتيبا فى كتابة عبارات IF THEN ELSE متعددة.

التطبيقات

عبارة SELECT CASE وسيلة مفيدة فى البرمجة المرتبة التى تقود الى برامج متماسكة وسهلة القراءة وتكون سهلة كذلك فى صيانتها. وفيما يلى امثلة لعبارات SELECT CASE .

```
SELECT CASE Choice$
     CASE IS "A"
          GOSUB AddRecords
     CASE "E" TO "H"
          GOSUB EditProces
     CASE ELSE
          GOSUB SurpriseProcess
END SELECT

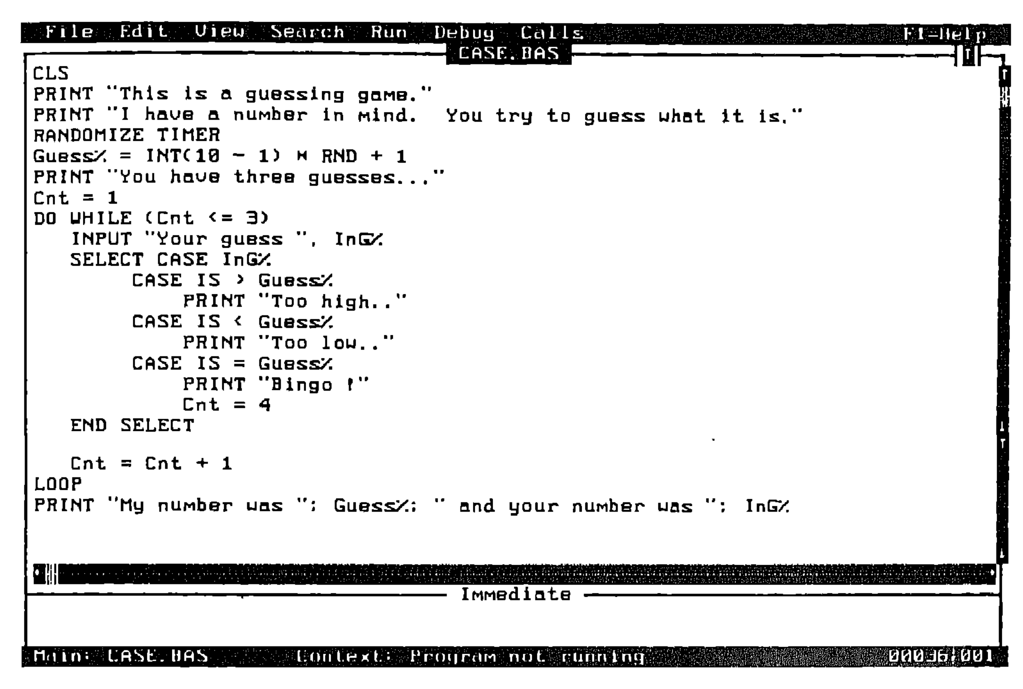
INPUT "Enter value: ";V%
SELECT CASE V%
     CASE IS > 10
          ..
     CASE 1 TO 3
          ..
END SELECT
```

عملية تقليدية

فى هذه العملية تقوم بتعديل البرنامج المقدم فى الدرس الثامن والخمسين لاستخدام عبارة SELECT CASE . ابدأ بتحميل بيسك السريع.

١ – من قائمة File اختر Open واضغط على Tab للذهاب الى الدليل . اختر IF THEN. BAS واضغط على مفتاح الادخال.

٢ – عدل عبارة IF THEN ELSE كما هو مبين فى القائمة التالية :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
CASE.BAS
CLS
PRINT "This is a guessing game."
PRINT "I have a number in mind.  You try to guess what it is."
RANDOMIZE TIMER
Guess% = INT(10 - 1) * RND + 1
PRINT "You have three guesses..."
Cnt = 1
DO WHILE (Cnt <= 3)
   INPUT "Your guess ", InG%
   SELECT CASE InG%
        CASE IS > Guess%
            PRINT "Too high.."
        CASE IS < Guess%
            PRINT "Too low.."
        CASE IS = Guess%
            PRINT "Bingo !"
            Cnt = 4
   END SELECT

   Cnt = Cnt + 1
LOOP
PRINT "My number was "; Guess%; " and your number was "; InG%

Immediate
Main: CASE.BAS        Context: Program not running        00036:001
```

٣ – نفذ البرنامج وادخل تخميناتك.

```
This is a guessing game.
I have a number in mind.  You try to guess what it is.
You have three guesses...
Your guess 3
Too low..
Your guess 5
Too low..
Your guess 12
Too high..
My number was  8  and your number was  12




Press any key to continue
```

٤ – لاحظ استخدام عبارة SELECT CASE فى البرنامج وكيفية تحسينه فى قراءة البرنامج.
اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٥ – سوف نحفظ هذا البرنامج فى صورته المعدلة مستخدما اسم ملف مختلف من قائمة File اختر Save As واحذف اسم الملف الاصلى بالضغط على Del ثم اكتب CASE.BAS كاسم للملف. حدد أن شكل الملف نصى واحفظ هذا البرنامج.

٦ – من قائمة File اختر New مع اخلاء الشاشة .

٧ – انتقل الى الدرس التاسع عشر للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع والعشرون

دالة SETMEM

الوصف

دالة SETMEM تزيد او تقلل من الذاكرة المستخدمة بواسطة الكومة البعيدة، والكومة البعيدة far heap هى المكان المخزن فيه الاشياء البعيدة وتكوين العبارة هو كما يلى :

```
SETMEM(number)
```

جزء number هو تعبير عددى يمكن ان يكون سالبا او موجبا . وعندما يكون موجبا تزداد الكومة البعيدة بعدد البايت المحدد، وعندما يكون العدد اكبر من الذاكرة المتاحة فتتحدد كل الذاكرة المتاحة. وتعيد الدالة كمية ذاكرة الكومة البعيدة المحددة بالبايت، وعندما يكون العدد سالبا فتقل الذاكرة بعدد البايت المحدد. فاذا كان العدد صفرا فتعود دالة SETMEM بالحجم الحالى للكومة البعيدة.

التطبيقات

تستخدم دالة SETMEM فى معظم الحالات فى البرمجة بخليط من اللغات، وفيما يلى مثال لدالة SETMEM .

```
DIM R(100)
..
Fh = SETMEM(0)
PRINT "Far heap = ";Fh;"bytes"
```

عملية تقليدية

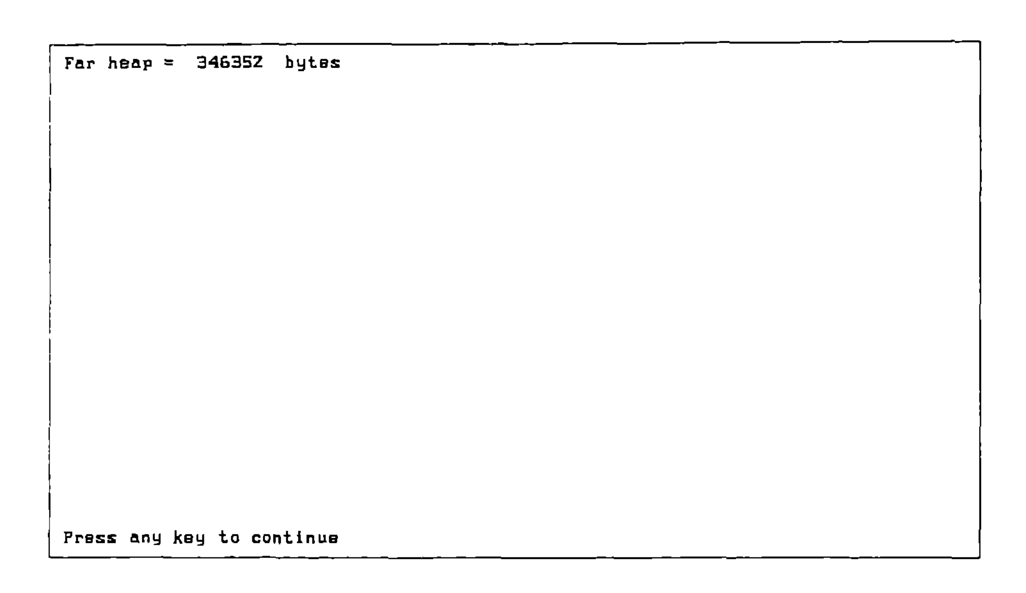
العملية التالية توضح دالة SETMTM فى صورة بسيطة . والبرنامج عبارة عن تنفيذ للمثال الموجود فى قسم التطبيقات، ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                               <Untitled>
'This program demonstrates the SETMEM function.
CLS
DIM T(100) AS STRING
'get the current size of the far heap
Fh = SETMEM(0)
PRINT "Far heap = "; Fh; " bytes"

                               Immediate
```

٢ – نفذ البرنامج. لاحظ نتيجة دالة SETMEM فى البرنامج.



٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والثامن والعشرون

دالة SGN

الوصف

دالة SGN تعيد اشارة التعبير العددى، وتكوينها هو كما يلى :

```
SGN(numeric expression)
```

نعطى دالة SGN احدى ثلاث نتائج ممكنة طبقا للتعبير العددى. والجدول التالى يسرد النتائج الممكنة .

النتيجة	السبب
1	التعبير العددى اكبر من 0 .
- 1	التعبير العددى اقل من 0 .
0	التعبير العددى مساويا 0 .

التطبيقات

دالة SGN تكون مفيدة عندما تكون هناك حاجة الى اشارة العدد لاتخاذ قرار معين فى البرنامج .(يمكن الحصول على نفس التاثير باختبار ما إذا كانت القيمة اكبر من او مساو لـــ او اقل من صفر). وفيما يلى امثلة تستخدم دالة SGN .

```
PRINT SGN(12-23)
IF SGN(BalanceDue!) = 1 THEN  GOTO CreditRecovery
PRINT SGN((2.1/3)-Srt%)
```

عملية تقليدية

توضح دالة SGN فى هذه العملية التقليدية . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'The program demonstrates the SGN function
x = 42: y = 10: z = -5
a = x - (y * z)
b = (y + z) * 0
c = (x / y) * z
PRINT a, "SGN(A) = "; SGN(a)
PRINT b, "SGN(B) = "; SGN(b)
PRINT c, "SGN(C) = "; SGN(c)

                          Immediate

Main: <Untitled>    Context: Program not running          00001:029
```

٢ - نفذ البرنامج ولاحظ استخدام دالة SGN فى البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

```
 92            SGN(A) =  1
 0             SGN(B) =  0
-21            SGN(C) = -1


Press any key to continue
```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والرابع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع والعشرون

عبارة SHARED

الوصف

تقدم عبارة SHARED اتصالاً بمتغيرات سبق توضيحها على مستوى الجزء إلى SUB و FUNCTION وتكوينها هو كما يلى :

```
SHARED var1 AS type, var2 AS type,...
```

اجزاء var1 و var2 هى اسماء متغيرات وهى متغيرات بيسك سريع صحيحة وتشمل المنظومات كذلك. جزء AS type يحدد نوع بيانات المتغير. يمكن ان يكون النوع من انواع البيانات البسيطة او الانواع التى يحددها المستفيد. وعندما يكون المتغير منظومة فيتبعه قوسان فارغان مثلما يلى :

```
var1(), var2(), ..
```

التطبيقات

يمكن ان تظهر عبارة SHARED داخل SUB او FUNCTION فقط ويمكن ان تقتسم متغيرات موضحة فى هذا الجزء فقط وليس مع مكتبة سريعة او اى جزء اخر. يسمح ذلك لـــ SUB او FUNCTION ان يستخدم هذه المتغيرات دون ان تمرر كمؤشرات. وفيما يلى امثلة لعبارة SHARED .

مثال١

```
SUB Fewer
   SHARED Old AS INTEGER, New AS INTEGER
   ..
END SUB
```

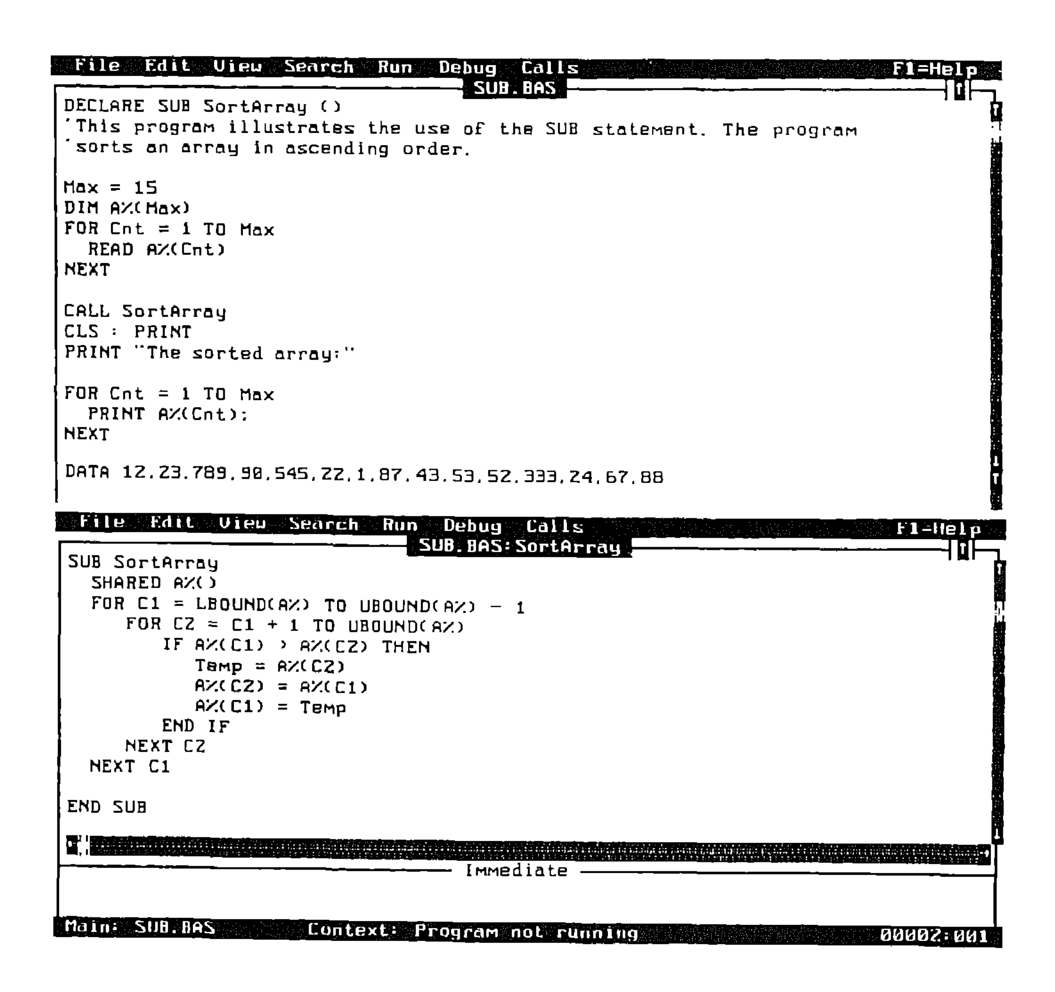
مثال٢

```
FUNCTION ConstStr$
     SHARED Char AS STRING
     ..
END FUNCTION
```

عملية تقليدية

هذه العملية توضح استخدام عبارة SHARED . ابدأ بتحميل بيسك السريع.

١ - حمل برنامج SUB.BAS الذى سبق اعداده فى الدرس المائة والرابع والأربعين وعدل البرنامج كما هو مبين فى القائمة التالية :



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              SUB.BAS
DECLARE SUB SortArray ()
'This program illustrates the use of the SUB statement. The program
'sorts an array in ascending order.

Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
  READ A%(Cnt)
NEXT

CALL SortArray
CLS : PRINT
PRINT "The sorted array:"

FOR Cnt = 1 TO Max
  PRINT A%(Cnt);
NEXT

DATA 12,23,789,90,545,22,1,87,43,53,52,333,24,67,88
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         SUB.BAS:SortArray
SUB SortArray
  SHARED A%()
  FOR C1 = LBOUND(A%) TO UBOUND(A%) - 1
     FOR C2 = C1 + 1 TO UBOUND(A%)
        IF A%(C1) > A%(C2) THEN
           Temp = A%(C2)
           A%(C2) = A%(C1)
           A%(C1) = Temp
        END IF
     NEXT C2
  NEXT C1

END SUB
                              Immediate
Main: SUB.BAS        Context: Program not running              00002:001
```

٢ - نفذ البرنامج . لاحظ استخدام عبارة SHARED فى جعل المنظومة متاحة لـ SUB.

مخرجات البرنامج هى كما يلى :

```
The sorted array:
 1  12  22  23  24  43  52  53  67  87  88  90  333  545  789




Press any key to continue
```

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثلاثون

عبارة SHELL

الوصف

تسمح لك عبارة SHELL بالخروج من البرنامج وتنفيذ امر DOS والعودة مرة اخرى الى البرنامج. وتكوينها هو كما يلى :

```
SHELL command
```

جزء command هو امر DOS صحيح مثلDIR او COPY ويوضع بين علامتى تنصيص. وهو جزء اختيارى وعندما تستخدم عبارة SHELL بدونه فإنها تأخذك الى DOS وتسمح لك بتنفيذ اى امر من اوامر DOS والعودة بعد ذلك الى البرنامج عندما تكتب "EXIT" . ومن الممكن تنفيذ برامج باستخدام هذه الوسيلة.

التطبيقات

عبارة SHELL مفيدة فى تنفيذ ملفات EXE . و COM. و BAT ، من البرنامج وفى تنفيذ انشطة DOS اخرى. وفيما يلى امثلة لعبارة SHELL :

مثال١

```
SHELL "DIR /W"
```

مثال٢

```
SHELL "TYPE TOC | SORT > TOC.SRT"
```

مثال٣

```
SHELL
```

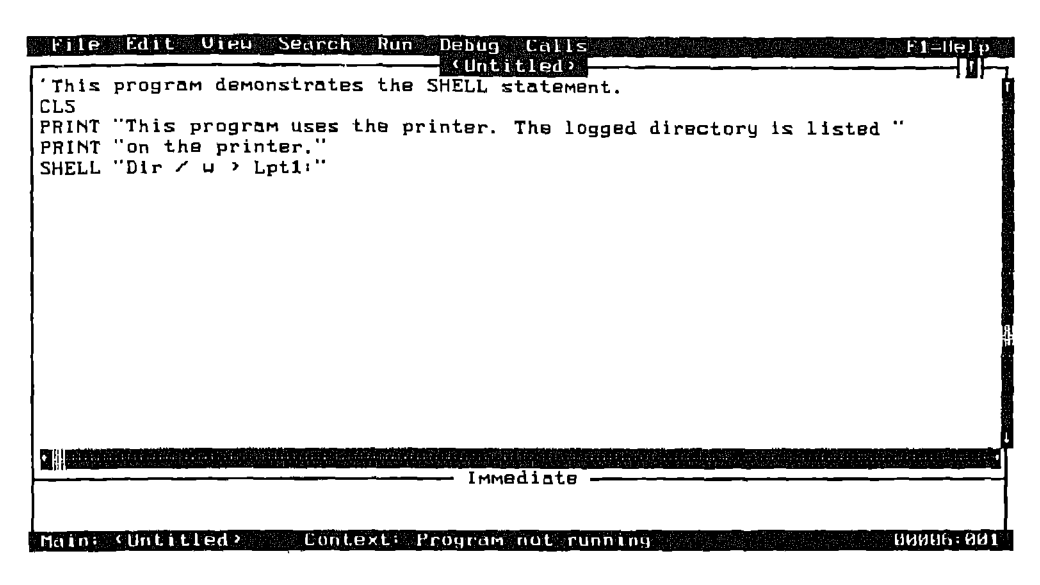
يخرج المثال الاول الى DOS ويعرض الملفات الموجودة فى الدليل المفتوح ويعود الى البرنامج. وينسخ المثال الثانى ملف TOC ويستخدم مرشح ترتيب DOS ويعيد ترتيب المخرجات الى ملف TOC. SRT آخر ثم يعيدك الى البرنامج . ويخرج المثال الثالث من DOS وينتظر امرا. يعود التحكم الى البرنامج عندما تكتب EXIT.

عملية تقليدية

هذا هو توضيح بسيط لعبارة SHELL. استمر اذا كان لديك طابع متصل بالكمبيوتر فقط.

ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج . عند التنفيذ تشبه الشاشة ما يلى : لاحظ استخدام عبارة SHELL فى البرنامج.

```
This program uses the printer. The logged directory is listed
on the printer.



Press any key to continue
```

٣ - ارجع الى البرنامج اخرج من بيسك السريع دون ان تحفظ البرنامج. بهذا تكون قد اتممت تسلسل التعلم. استمر فى الملاحق للمزيد من المعلومات.

الدرس المائة وواحد وثلاثون

دالة SIN

الوصف

تحسب دالة SIN جيب الزاوية المعطاة بالتقدير الدائرى. وتكوينها هو كما يلى :

```
SIN(numeric expression)
```

قيمة SIN تحسب بدقة فردية كقيمة تقليدية، وعندما يكون التعبير العددى بدقة مزدوجة فتحسب قيمة SIN بدقة مزدوجة . ويمكن ان يكون التعبير العددى اى عدد.

التطبيقات

تستخدم دالة SIN عندما يراد حساب جيب الزاوية كما فى حالة رسم الرسومات. وفيما يلى بعض امثلة لدالة SIN .

```
PRINT SIN(12.22)
T = SIN(.03)
PRINT T
```

عملية تقليدية

العملية التالية توضح دالة SIN ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
'This program demonstrates the SIN function
'The program computes a simulated sine value and compares it
'with the standard SIN function. The simulated sine is computed thus:
'sin x = x - x^3/3! + x^5/5! + x^7/7! + ...
CLS
PRINT : PRINT "Simulated Sine value and built-in SIN function result"
PRINT : PRINT "x = .1"
x = .1: n = 1: t = x: s = x
```

```
Cont:
n = n + 2
t = (-t * x ^ 2) / (n * (n - 1))
s = s + t
IF ABS(t) <= .00001 THEN GOTO Cont
PRINT "Simulated sine x "; s
PRINT "Built-in SIN function result "; SIN(x)

Immediate

Main: <Untitled>          Context: Program not running          00013:013
```

٢ - نفذ البرنامج ولاحظ استخدام دالة SIN فى البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

```
Simulated Sine value and built-in SIN function result

x = .1
Simulated sine x  8.333334E-02
Built-in SIN function result  9.983342E-02

Press any key to continue
```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والخامس والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثانى والثلاثون

عبارة SOUND

الوصف

تنتج عبارة SOUND صوتا بتردد محدد وديمومة محددة من الكمبيوتر. وتكوينها هو كمايلى :

```
SOUND frequency,duration
```

التردد عبارة عن تعبير عددى يقع فى المدى من 37 الى 32,767. والديمومة هى طول وقت استمرارية الصوت الناتج وهو تعبير عددى يقع فى المدى من 0 الى 65,535 . ويفسر التردد بانه عدد من الدورات فى الثانية وتفسر الديمومة بأنها دورات الساعة الداخلية للكمبيوتر. وهناك 18.2 دورة للساعة فى الثانية الواحدة.

التطبيقات

تستخدم عبارة SOUND فى انتاج صوت من الكمبيوتر بطرق عديدة . وفيما يلى بعض الامثلة :

```
SOUND 500, 500
SOUND T%*2, T%*100
SOUND 100,0
```

اخر مثال له ديمومة تساوى صفراً ويتسبب ذلك فى ان يتوقف متحدث الكمبيوتر.

عملية تقليدية

هذه العملية توضح استخدام عبارة SOUND. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                         F1=Help
                              <Untitled>
'This program demonstrates the SOUND statement
CLS
PRINT : PRINT
PRINT " This program will produce a sound depending on what key you type."
PRINT " Use the number keys. Type 0 to quit."
C$ = ""
DO WHILE (C$ <> "0")
   LOCATE 6, 1: INPUT C$
   IF (C$ <> "0") THEN
     Snd% = VAL(C$)
     IF Snd% < 9 THEN SOUND Snd% * 120, 25

   END IF

LOOP

'Turn the sound off
SOUND 0, 0
                              Immediate

Main: <Untitled>      Context: Program not running            C  00018:011
```

٢ – نفذ البرنامج. لاحظ تأثير عبارة SOUND فى البرنامج. اكتب اى رقم واضغط على مفتاح الادخال. (الرقم 9 يقع خارج مدى سماع البشر). اكتب 0 واضغط على مفتاح الادخال لايقاف البرنامج.

```
This program will produce a sound depending on what key you type.
Use the number keys. Type 0 to quit.

? 2
```

٣ – من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ – انتقل الى الدرس المائة وثلاثة للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والثلاثون

دالة SPACE$

الوصف

هذه الدالة تعيد سلسلة فراغات لها طول معين. وتكوينها هو كما يلى :

```
SPACE$(numeric expression)
```

يجب ان يقع التعبير العددى فى المدى من 0 الى 32,767.

التطبيقات

تستخدم دالة SPACE$ عندما تكون هناك حاجة الى سلسلة فراغات . ويستخدم البرنامج العينة الموجود فى الدرس الثالث دالة SPACE$ فى تحديد وضع جانى الصندوق على الشاشة. وفيما يلى امثلة اخرى لاستخدام دالة SPACE$.

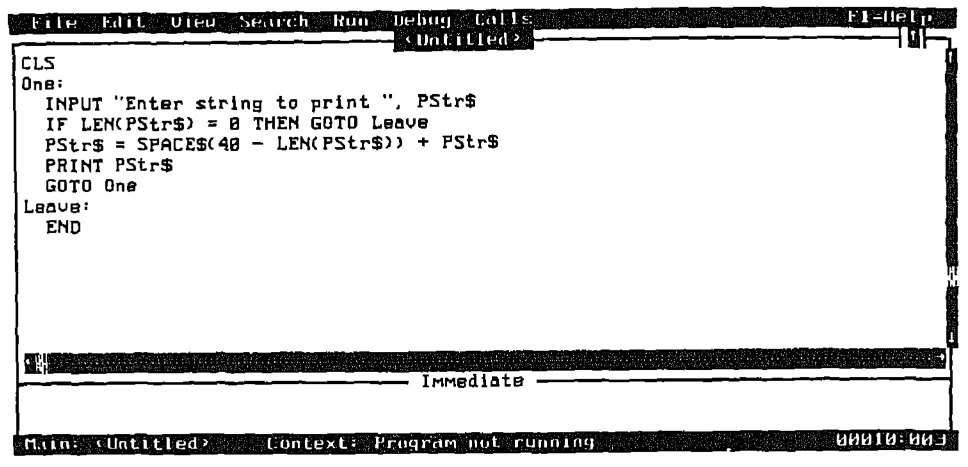
```
PRINT 22 SPACE$(22) 22
22                      22
PRINT "'hello" SPACE$(40) "there!"
'hello
```

نصيحة : يمكن ان تستخدم دالتا TAB و SPC فى تحديد وضع المخرجات اثناء استخدام عبارة PRINT .

عملية تقليدية

يجرى البرنامج التالى تجاربا مع دالة SPACE$. ولاختباره ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
CLS
One:
  INPUT "Enter string to print ", PStr$
  IF LEN(PStr$) = 0 THEN GOTO Leave
  PStr$ = SPACE$(40 - LEN(PStr$)) + PStr$
  PRINT PStr$
  GOTO One
Leave:
  END
```

٢ - اضغط على Shift - F5 لتنفيذ البرنامج. اكتب اسطرا من Mary Had a Little Lamb او من اى مقطوعة اخرى كرد على "Enter string to print" بعد ثلاثة أو اربعة اسطر اضغط ببساطة على مفتاح الادخال لإنهاء البرنامج.

```
Enter string to print Mary had a little lamb.
                  Mary had a little lamb.
Enter string to print Its fleece was white as snow
            Its fleece was white as snow
Enter string to print And everywhere that Mary went
          And everywhere that Mary went
Enter string to print The lamb was sure to go !
               The lamb was sure to go !
Enter string to print
```

لاحظ كيف يقبل البرنامج مدخلات على هيئة سلسلة مع ملء الاماكن الزائدة الموجودة على اليسار بفراغات وطباعة السلسلة . ويتسبب ذلك فى طباعة السلسلة مرحلة ناحية اليمين. ونشجعك على اجراء تعديلات على الشفرة واجراء تجارب .

٣ - اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٤ - اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لازالة البرنامج.

٥ - انتقل الى الدرس التاسع والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع والثلاثون

دالة SPC

الوصف

تترك دالة SPC عدداً محدداً من الفراغات في عبارة PRINT او LPRINT . وتكوينها هو كما يلى :

```
SPC(numeric expression)
```

ويجب ان يكون التعبير العددى واقعا فى المدى من 0 الى 32,767 .

التطبيقات

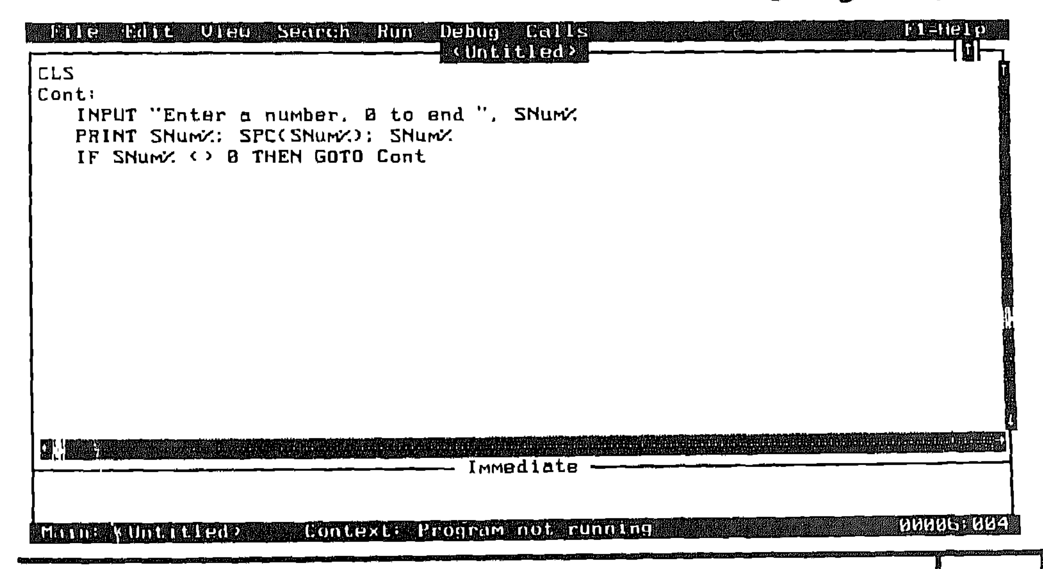
يمكن ان تستخدم دالة SPC مع عبارات PRINT و LPRINT. وفيما يلى بعض الامثلة .

```
PRINT SPC(10) "Here" SPC(10) "Here" SPC(20) "And Here"
PRINT "Minimum value: " SPC(5) Min%
LPRINT "Name" SPC(30) "Address": LPRINT STRING$(41,"-")
```

عملية تقليدية

البرنامج التالى يوضح دالة SPC. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :



٢ – اضغط على Shift - F5 لتنفيذ البرنامج واضغط على ارقام لاختباره.

```
Enter a number, 0 to end 12
 12               12
Enter a number, 0 to end 22
 22                        22
Enter a number, 0 to end 33
 33                                   33
Enter a number, 0 to end 45
 45                                               45
Enter a number, 0 to end 1
 1   1
Enter a number, 0 to end 22
 22                        22
Enter a number, 0 to end 0
 0  0




Press any key to continue
```

٣ – اضغط على اى مفتاح للعودة الى البرنامج. اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لاخلاء الشاشة.

٤ – انتقل الى الدرس السادس للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس والثلاثون

دالة SQR

الوصف

تعيد دالة SQR الجذر التربيعى لعدد معين. وتكوينها هو كما يلى :

```
SQR(numeric expression)
```

التعبير العددى هو عدد اكبر من او يساوى صفراً.

التطبيقات

تستخدم دالة SQR عندما يراد حساب الجذر التربيعى لعدد مثل ما يحدث فى التطبيقات الرياضية والرسومات. وفيما يلى مثال لدالة SQR:

```
FOR T = -3 TO +3 STEP .50
    PRINT SQR(ABS(T))
NEXT T
```

لاحظ ان دالة ABS مستخدمة فى التأكد ان التعبير العددى موجب.

عملية تقليدية

العملية التالية توضح استخدام دالة SQR. ابدا بتحميل بيسك السريع .

١ – اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                     F1=Help
                          <Untitled>
'This program solves simple quadratic equations.
CLS
INPUT "Enter a,b,c "; A, B, c
d = B ^ 2 - 4 * A * c
IF d < 0 THEN GOTO PrintMsg

E1 = (-B + SQR(d)) / (2 * A)
E2 = (-B - SQR(d)) / (2 * A)

PrintLine:
  PRINT "A = "; A; " B = "; B; " C = "; c
  PRINT "Root 1 = "; E1, "Root 2 = "; E2

END
PrintMsg:
 PRINT "Results are complex, and thus ignored."

                          Immediate

Main: <Untitled>     Context: Program not running          00016:040
```

٢ – نفذ البرنامج، اكتب 1 على انه a و 2 على انه b و 1- على انه c. لاحظ استخدام دالة SQR في البرنامج. اضغط اى مفتاح للعودة الى البرنامج.

```
Enter a,b,c ? 1,2,-1
A =  1  B =  2  C = -1
Root 1 =  .4142135          Root 2 = -2.414214





Press any key to continue
```

٣ – من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ – انتقل الى الدرس المائة والثامن والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والسادس والثلاثون

عبارة STATIC

الوصف

تستخدم عبارة STATIC فى توضيح متغيرات ومنظومات محلية لـ DEF FN او FUNC-TION او SUB لحفظ قيم هذه المتغيرات بين الاستدعاءات ، وتكوينها هو كما يلى :

```
STATIC variable list
```

جزء variable list له الشكل التالى :

```
var1 ( ) AS type, var2 ( ) AS type
```

تستخدم الاقواس اذا كان اى من var1 او var2 منظومة وتحديد الابعاد اختيارياً ، جزء AS type يعطى نوع المتغير ويمكن ان يكون من اى نوع من الانواع البسيطة او التى يحددها المستفيد.

تستخدم عبارة STATIC فى عبارة SUB او FUNCTION او DEF FN فقط ، والمتغيرات الموضحة على انها استاتيكية STATIC فى الدوال لها أولوية داخل المتغيرات الشاملة بنفس الاسم اذا ما وجد مثل ذلك. والثوابت الموضحة فى عبارة STATIC تعامل بنفس الطريقة ، وعندما توضح عبارة STATIC أياً من SUB او FUNCTION او DEF FN فتعامل كل المتغيرات المحتواة داخله على أنها محلية.

التطبيقات

تستخدم عبارة STATIC اساسا فى التحكم فى الاتصال بالمتغيرات المستخدمة داخل دالة معينة او برنامج فرعى معين. ويمكن ان تستخدم على مستوى توضيح دالة وتتسبب فى جعل المتغيرات محلية وتحفظ قيمها بين الاستدعاءات او يمكن ان تستخدم داخل الدالة او البرنامج الفرعى لحفظ قيم متغيرات معينة بين الاستدعاءات ، وفيما يلى بعض الامثلة :

مثال١

```
DEF FNCountRec(TRec) STATIC
     Count  = Count + 1
     ..
END DEF
```

مثال٢

```
SUB MoveContents(CCrec)
     STATIC MvCnt AS INTEGER, MvErr AS INTEGER
     DIM WriteRec AS CRec
     WriteRec.Fldl = CCrec.Fldl
     ..
END SUB
```

عملية تقليدية

العملية توضح استخدام عبارة STATIC، ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                        F1=Help
                          <Untitled>
'This program demonstrates the use of the STATIC statement.
'The program asks for a filename and counts the number of lines
'in that file. The file must be a text file.

DECLARE SUB NoOfLines (FileName$)

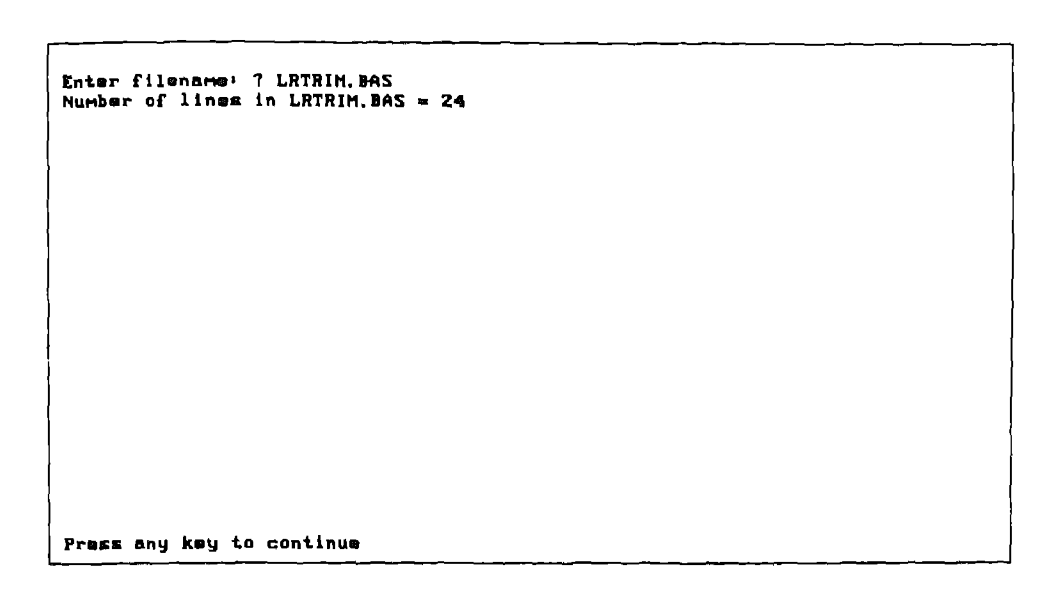
CLS
PRINT : INPUT "Enter filename: ": FileName$
IF FileName$ < > "" THEN
   CALL NoOfLines(FileName$)
   PRINT "Number of lines in "; FileName$; " =": LineCnt%
ELSE
END IF
 File  Edit  View  Search  Run  Debug  Calls                        F1=Help
                     <Untitled>:NoOfLines
SUB NoOfLines (FileName$) STATIC
   SHARED LineCnt%, line$
   OPEN FileName$ FOR INPUT AS #1
   LineCnt% = 0
   DO WHILE NOT EOF(1)
     LINE INPUT #1, line$
     LineCnt% = LineCnt% + 1
   LOOP

END SUB

                           Immediate

Main: <Untitled>      Context: Program not running                 00009:004
```

٢ – نفذ البرنامج. اكتب اسم ملف نص (LRTRIM.BAS على سبيل المثال) واضغط على مفتاح الادخال . لاحظ استخدام STATIC فى البرنامج.



```
Enter filename: ? LRTRIM.BAS
Number of lines in LRTRIM.BAS = 24

Press any key to continue
```

٣ – اختر New مع اخلاء الشاشة دون ان تحفظ البرنامج.

٤ – انتقل الى الدرس الثالث والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع والثلاثون

اشباه الاوامر $STATIC و $DYNAMIC

الوصف

تستخدم اشباه الاوامر $STATIC و $DYNAMIC فى توضيح منظومات . والمنظومة الاستاتيكية تكون لها ذاكرة محددة لها اثناء ترجمة البرنامج. اما المنظومة الديناميكية فتكون لها ذاكرة محددة لها اثناء تنفيذ البرنامج فقط. وعندما تستخدم اشباه الاوامر هذه بمهارة فإنها تقدم طريقة قوية للتحكم فى كيفية ادارة ذاكرة وقت تنفيذ البرنامج . وهذه الامكانية مهمة عندما يتطلب البرنامج كمية ذاكرة كبيرة وتكون بيئة التطوير او بيئة المقصد لها كمية ذاكرة متاحة اقل من اللازم. والتكوين هو كما يلى :

```
REM $STATIC
REM $DYNAMIC
```

تقدم اشباه الاوامر فى عبارة تعليق . يمكن استبدال جزء REM بفاصلة. ولا تكون هناك حاجة دائمة الى توجيهات. كما ان المنظومات تقسم ضمنيا الى ديناميكية واستاتيكية اعتمادا على كيفية توضيحها . ارجع الى الدرس الثالث والثلاثين لمناقشة كيفية توضيح منظومة بأنها استاتيكية او ديناميكية.

التطبيقات

تستخدم $STATIC و $DYNAMIC فى تحسين ادارة ذاكرة وقت التنفيذ . فمع بعض البرامج يعد مكان اكبر للسلاسل عن طريق استبدال المنظومات الاستاتيكية بمنظومات ديناميكية . العبارات الاخرى التى تسخدم مع اشباه الاوامر هذه هى عبارات DIM و REDIM و ERASE . وفيما يلى مثال لذلك :

```
REM $STATIC
CONST X1 = 20, Y1 = 80, X2 = 10, Y2 = 12
DIM WindowTwo(X1,Y1,X2,Y2)
'$DYNAMIC
DIM WindowOne(80,25)
..
REDIM WindowOne(40,25)
..
ERASE WindowOne
```

فى هذا المثال المنظومة WindowTwo هى منظومة استاتيكية. ويجب ان تكون استاتيكية حتى بدون اشباه الاوامر وذلك لاستخدام الثوابت فى الابعاد. اما المنظومة WindowOne فهى ديناميكية ويعاد اعداد ابعادها فيما بعد فى البرنامج. اكثر من هذا فإنها موضحة وهذا يخلى الذاكرة التى تستخدمها المنظومة.

عملية تقليدية

هذه العملية توضح استخدام اشباه الاوامر $STATIC و $DYNAMIC فى البرنامج . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                           <Untitled>
'This program demonstrates the use of $STATIC and $DYNAMIC
'statements. The program declares arrays and deallocates them.
'The memory available before and after is displayed on the screen.

REM $STATIC
DIM W(100, 100)
REM $DYNAMIC
DIM Q(10000)
CLS
PRINT "Amount of free memory ";
PRINT FRE(-1)
REDIM Q(1000)
PRINT "After redimensioning Q ";
PRINT FRE(-1)
ERASE Q
PRINT "After erasing Q ";
PRINT FRE(-1)

                             Immediate

Main: <Untitled>     Context: Program not running              00017:019
```

٢ - نفذ البرنامج. لاحظ استخدام اشباه الاوامر $STATIC و $DYNAMIC فى البرنامج.

```
Amount of free memory  262970
After redimensioning Q  298970
After erasing Q  302986

Press any key to continue
```

٣ – ارجع الى البرنامج. احفظ هذا البرنامج كملف نص تحت اسم STA_DYN.BAS مع اخلاء الشاشة.

٤ – انتقل الى الدرس التاسع والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثامن والثلاثون

دالة STICK

الوصف

تستخدم دالة STICK فى قراءة احداثيات اثنين من عصا الالعاب joysticks. وتكوينها هو كما يلى :

```
STICK(n)
```

حيث n هى رقم صحيح يقع بين 0 و 3. والقيم التى تعيدها الدالة لكل قيمة من قيم n هى كما يلى :

N	القيمة التى تعود
0	احداثى X لعصا الالعاب A.
1	احداثى Y لعصا الالعاب A عند اخر (0) STICK .
2	احداثى X لعصا الالعاب B .
3	احداثى Y لعصا الالعاب B عند اخر (0) STICK .

التطبيقات

دالة STICK تستخدم خصيصا مع عصا الالعاب . ويجب استدعاء (0) STICK قبل استدعاء اى دالة من دوال (n) STICK وذلك لأن (0) STICK تسجل إحداثيات عصا الالعاب الاخرى واحداثى X لعصا الالعاب الاولى .

انتقل الى الدرس المائة والحادى والأربعين للاستمرار فى تسلسل التعلم .

الدرس المائة والتاسع والثلاثون

عبارة STOP

الوصف

تتسبب عبارة STOP فى انهاء تنفيذ البرنامج. وتكوينها هو كما يلى :

```
STOP
```

يمكن استخدام عبارة STOP فى اى مكان فى البرنامج لانهاء البرنامج. وعندما تستخدم عبارة STOP فى بيئة تطوير بيسك السريع فإنها تنهى البرنامج دون ان تغلق اى ملف ولا تعود الى نظام التشغيل. وعندما تسخدم عبارة STOP فى برنامج قائم بذاته فإنها تغلق كل الملفات وتنهى البرنامج وتعود الى نظام التشغيل. وعندما يتم ترجمة البرنامج كبرنامج قائم بذاته بخيار d/ او x/ او e/ فتطبع عبارة STOP رقم السطر الاقرب ما يمكن الى السطر الذى انتهى عنده البرنامج. وعندما لا توجد اى ارقام اسطر فى البرنامج فتطبع عبارة STOP صفرا.

التطبيقات

تستخدم عبارة STOP لانهاء البرنامج طبقا لتمييز المبرمج. وفيما يلى بعض الامثلة :

مثال١

```
INPUT "Enter file name ":FileName$
..
STOP
..
```

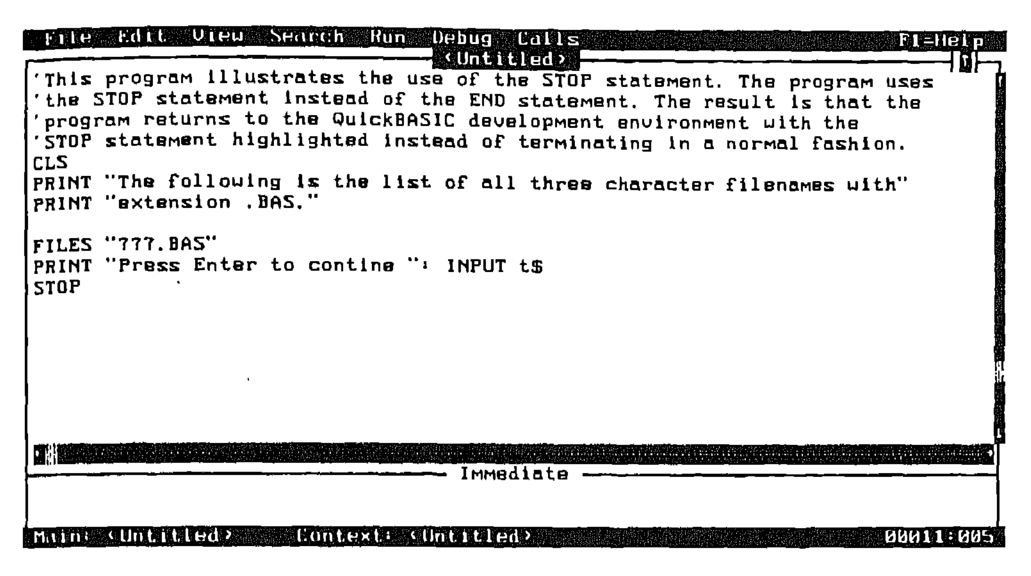
مثال٢

```
FOR T = 1 TO 20 : PRINT (X•12.2): NEXT T
STOP
```

عملية تقليدية

هذه العملية توضح عبارة STOP . ابدأ بتحميل بيسك السريع .

١ – اكتب البرنامج التالى :



٢ - نفذ البرنامج. اجب على الملقنات التى تظهر على الشاشة. لاحظ استخدام عبارة STOP فى البرنامج لاحظ كذلك كيفية اعادة البرنامج الى بيسك السريع مع زيادة اضاءة عبارة STOP.

```
The following is the list of all three character filenames with
extension .BAS.
C:\QB
BOX     .BAS     ASC     .BAS     ABS     .BAS
 872448 Bytes free
Press Enter to contine
?
```

```
File  Edit  View  Search  Run  Debug  Calls                           F1=Help
                                <Untitled>
'This program illustrates the use of the STOP statement. The program uses
'the STOP statement instead of the END statement. The result is that the
'program returns to the QuickBASIC development environment with the
'STOP statement highlighted instead of terminating in a normal fashion.
CLS
PRINT "The following is the list of all three character filenames with"
PRINT "extension .BAS."

FILES "???.BAS"
PRINT "Press Enter to contine "; INPUT t$
STOP

                                 Immediate

Main: <Untitled>        Context: <Untitled>                       00011:001
```

٣ - انتقل الى البرنامج مع اخلاء الشاشة دون ان تحفظ البرنامج.

٤ - انتقل الى الدرس المائة والسادس والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والأربعون

دالة STR$

الوصف

تحول دالة STR$ تعبيرا عدديا الى ما يمثله من سلسلة، ومتمم دالة STR$ هو دالة VAL والتى تحول سلسلة الى عددها المكافىء لها. وتكوينها هو كما يلى :

```
STR$(numeric expression)
```

يتحول العدد الى السلسلة التى تمثله رقما رقما. اذا ما احتوى التعبير العددى على نقطة فإنها تتحول كذلك. واذا كانت قيمة التعبير العددى موجبة فتحتوى السلسلة على فراغات فى بدايتها وإلا فإنها تحتوى على اشارة سالب كأول رمز لها.

التطبيقات

دالة STR$ هى دالة مريحة عندما تكون هناك حاجة الى تشكيل طباعة التعبير العددى. ويمكن حذف الفراغات التى توجد فى البداية او النهاية وتصبح المخرجات مضغوطة وسهلة القراءة. وفيما يلى بعض الامثلة :

```
X% = 2777
PRINT STR$(X%)
```

المخرجات : 2777

```
PRINT STR$(X% / 3)
```

المخرجات : 925.6667

```
PRINT STR$((X% / 3.1) * -1)
```

المخرجات : 895.8065-

عملية تقليدية

استخدام دالة STR$ موضح فى هذه العملية. ابدأ بتحميل بيسك السريع.

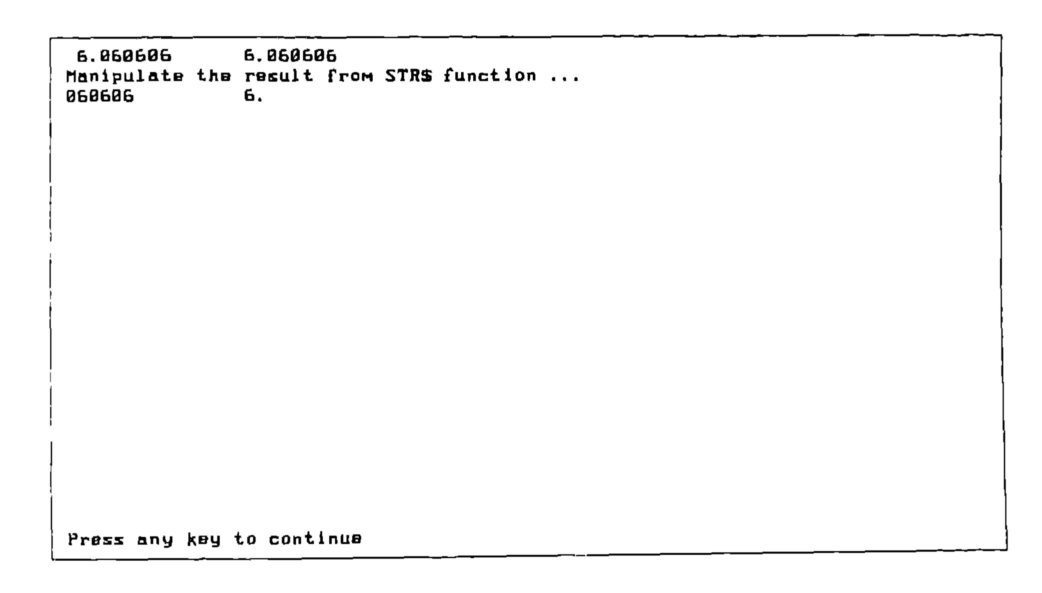
١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This is a demonstration of the STR$ function.
CLS
IVal = 20
TVal = 3.3
UVal = IVal / TVal
SVal$ = STR$(UVal)
PRINT UVal, SVal$
PRINT "Manipulate the result from the STR$ function ..."
PRINT RIGHT$(SVal$, 6), LEFT$(SVal$, 3)

                              Immediate

Main: <Untitled>      Context: Program not running              00009:040
```

٢ – نفذ البرنامج، لاحظ المخرجات واستخدام دالة STR$ فى البرنامج، تتحول القيمة العددية VVal الى السلسلة (SVal$) وتنقسم الى جزءين وتطبع .



٣ – اضغط على اى مفتاح للعودة الى البرنامج، اضغط على Alt - F واضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ – انتقل الى الدرس السابع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة الحادى والأربعون

دالة STRIG

الوصف

تعيد دالة STRIG حالة اطلاق عصا العاب معينة ، وتكوينها هو كما يلى :

```
STRIG(n)
```

الجزء n هو قيمة عددية صحيحة تقع بين 0 و 7 . ويصف الجدول التالى نتائج كل قيمة من قيمn.

N	النتيجة
0	اذا كان الزر السفلى لعصا اللعب مضغوطا منذ اخر STRIG (0) فتكون النتيجة 1 - وإلا فانها تكون 0 .
1	اذا كان الزر السفلى لعصا اللعب لاسفل حاليا عند A تكون النتيجة 1 - وإلا فهى 0
2	اذا كان الزر السفلى مضغوطا عند B منذ اخر STRIG (2) فتكون 1 - وإلا فهى 0.
3	اذا كان الزر السفلى لعصا اللعب على B لاسفل حاليا فتكون 1 - وإلا فهى 0.
4	اذا كان الزر العلوى لعصا اللعب على A مضغوطا منذ اخر STRIG (4) فتكون 1- وإلا فهى 0 .
5	اذا كان الزر العلوى لعصا اللعب على A مضغوطا حاليا فتكون 1 - وإلا فهى0.
6	اذا كان الزر العلوى لعصا اللعب على B مضغوطا منذ أخر STRIG (6) فتكون 1 - وإلا فهى 0 .
7	اذا كان الزر العلوى لعصا اللعب على B مضغوطا حاليا فتكون 1 - وإلا فهى 0 .

التطبيقات

تستخدم دالة STRIG فى الحصول على معلومات صريحة عن اثنين من عصا الالعاب.

انتقل الى الدرس المائة والثانى والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والثانى والأربعون

عبارات STRIG ON و STRIG OFF و STRIG STOP

الوصف

تقوم عبارات STRIG ON و STRIG OFF و STRIG STOP بتمكين والغاء تمكين وايقاف اصطياد الاحداث على عصا لعب محددة. وتكوين هذه العبارات كما يلى :

```
STRIG(n) ON
STRIG(n) OFF
STRIG(n) STOP
```

قيمة n هى احدى القيم 0 أو 2 أو 4 أو 6 . وتأثير كل قيمة من قيم n موجود فى الجدول التالى:

N	التأثيــــر
0	اصطياد الزر السفلى على عصا اللعب A.
2	اصطياد الزر السفلى على عصا اللعب B .
4	اصطياد الزر العلوى على عصا اللعب A.
6	اصطياد الزر العلوى على عصا اللعب B.

ويمكن عمل اصطياد احداث باستخدام عبارة STRIG (n) ON لزر عصا لعب محددة. واستخدام ON STRIG (n) GOTO : GOSUB يقوم بتشغيل الحدث كما يحدث.

وتلغى STRIG (n) OFF مقدرة اصطياد الاحداث لزر عصا لعب محددة. والاحداث التى تحدث بعد تنفيذ هذه العبارة لا يمكن تذكرتها.

وتوقف STRIG (n) STOP اصطياد الاحداث لزر عصا لعب محددة. والاحداث التى تحدث بعد ذلك يمكن تذكرتها وتشغيلها عندما تنفذ عبارة STRIG (n) ON بعد ذلك لهذا الزر.

التطبيقات

عبارات STRIG ON و STRIG OFF و STRIG STOP تكون اكثر فائدة عند كتابة برنامج يدعم المدخلات باستخدام عصا اللعب. وللمزيد من المعلومات عن اصطياد الاحداث ارجع الى الدرس الثانى والتسعين.

انتقل الى الدرس الحادى والستين للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والأربعون

دالة STRING$

الوصف

هذه الدالة تعيد سلسلة لها طول محدد تعد لرمز معين من رموز ASVII، وتكوينها هو كما يلي :

```
STRING$(L,c)
STRING$(L,string expression)
```

حيث L هو تعبير عددى تقع قيمته بين 1 و 32,767 يحدد طول السلسلة المطلوبة، و C هو تعبير عددى تقع قيمته بين 0 و 255 للرمز المراد استخدامه في بناء السلسلة. وتعبير السلسلة هو أى سلسلة ، ويستخدم اول رمز من تعبير السلسلة فى اعدادها.

التطبيقات

دالة STRING$ تكون اكثر فائدة فى بناء سلاسل عديدة لرموز مختلفة واطوال مختلفة، وأحد مثل هذه الاستخدامات موضح فى عينة البرنامج الموجودة فى الدرس الثالث حيث استخدمت الدالة فى بناء الاسطر العلوية والسفلية للصندوق. وفيما يلى امثلة اخرى لاستخدام دالة STRING $.

مثال١

```
PRINT STRING$(40,".")
........................................
```

مثال٢

```
PRINT STRING$(40,42)
........................................
```

مثال٣

```
PRINT "No! " STRING$(4,46) "Don't do that!"
No! .... Don't do that!
```

مثال٤

```
PRINT STRING$(8,"Sure")
SSSSSSSS
```

عملية تقليدية

يقبل البرنامج التالى عشر قيم لأحد الرسومات ويخزنها فى منظومة ويرسم الرسم طبقا لهذه القيم، وتستخدم دالة STRING$ فى رسم اسطر الرسم الافقية بمجهود بسيط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          STRING.BAS
DIM GU%(10)
GETINPUT:
INPUT "ENTER NEXT VALUE FOR GRAPH. 0 TO END ", I%
IF I% = 0 THEN GOTO DRAWGRAPH
GUS% = GUS% + 1: GU%(GUS%) = I%
IF (GUS% < 11) THEN GOTO GETINPUT
DRAWGRAPH:
CLS
PRINT STRING$(10, "-"); "YOUR GRAPH"; STRING$(10, "-"): PRINT : PRINT
FOR I = 1 TO 10: PRINT I; STRING$(GU%(I), "*"): NEXT

                          Immediate

Main: STRING.BAS      Context: Program not running         00001:001
```

٢ - اضغط على Shift - F5 لتنفيذ البرنامج.

```
ENTER NEXT VALUE FOR GRAPH, 0 TO END 12
ENTER NEXT VALUE FOR GRAPH, 0 TO END 2
ENTER NEXT VALUE FOR GRAPH, 0 TO END 22
ENTER NEXT VALUE FOR GRAPH, 0 TO END 44
ENTER NEXT VALUE FOR GRAPH, 0 TO END 55
ENTER NEXT VALUE FOR GRAPH, 0 TO END 0
```

```
------------YOUR GRAPH------------


 1 ************
 2 **
 3 **********************
 4 ********************************************
 5 *******************************************************
 6
 7
 8
 9
 10









Press any key to continue
```

٣ - اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٤ - من قائمة File اختر Save واكتب STRING.BAS كاسم للملف واحفظ هذا البرنامج.

٥ - انتقل الى الدرس المائة والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع والأربعون

عبارتا SUB و END..SUB

الوصف

تعرف عبارتى SUB وEND..SUB برنامج فرعيا . وفيما يلى تكوينها :

```
SUB name (parameter list)
END SUB
```

جزء name هو اسم البرنامج الفرعى المستخدم فى استدعاء البرنامج الفرعى. وجزء par- ameter list اختيارى وله التكوين التالى :

```
variable () AS type, variable () AS type
```

تحدد الاقواس اذا ما كان المتغير منظومة (وهذا اختيارى) فلا تكون هناك حاجة إلى الابعاد. ويحدد جزء AS type نوع البيانات التى ينتمى اليها المتغير (وهى اختيارى كذلك). ويمكن للنوع المحدد ان يكون من البيانات البسيطة او من البيانات التى يعرفها المستفيد.

وتمرر المؤشرات بواسطة دليل، واى تغيير يجرى على المؤشرات يؤثر على المتغيرات الاصلية. وللمزيد من المعلومات عن آلية تمرير المؤشر ارجع الى الدرس الثلاثين.

وتنهى END SUB البرنامج الفرعى. ويمكن الخروج من البرنامج الفرعى بصورة نهائية باستخدام عبارة EXIT SUB .

اعتبارات أخرى : فيما يلى قائمة بأشياء أخرى يجب تذكرها عن عبارات SUB.

- على عكس DEF FN او FUNCTION فلا يمكن استخدام SUB فى احد التعبيرات.
- يمكن للبرامج الفرعية SUB ان تتسم بسمة الاعادة الذاتية. فيمكنها ان تستدعى نفسها بنفسها.
- لا يمكن للبرامج الفرعية SUB ان تتداخل.

- لا يمكن للبرامج الفرعية SUB أن تحتوى على عبارات DEF FN او FUNCTION .
- لحفظ قيم المتغيرات المستخدمة فى برنامج فرعى SUB بين الاستدعاءات فتستخدم عبارة STATIC.
- تعتبر كل المتغيرات والمنظومات محلية للبرنامج الفرعى إلا إذا ما عرفت بصورة محددة بأنها مشتركة باستخدام عبارة SHARED.

التطبيقات

تستخدم عبارة SUB بنفس الطريقة مثل DEF FN فى عزل شفرة لها غرض محدد تحديدا جيدا وخالية من الخطأ . وهناك مميزات اضافية لتكوين SUB و END SUB كما سبق توضيحه فى هذا الدرس. وفيما يلى بعض الامثلة.

مثال١

```
SUB NextRec(CustRec)
     ..
     IF EOF(#2) THEN EXIT SUB
END SUB
```

مثال٢

```
SUB WriteRec(CustRec) STATIC
     Count = Count + 1
     ..
END SUB
```

يوضح هذا المثال استخدام STATIC فى عبارات SUB . ويتسبب ذلك فى ان المتغير count يحتفظ بقيمته السابقة بين الاستدعاءات .

عملية تقليدية

هذه العملية توضح استخدام عبارة SUB. ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This program illustrates the use of the SUB statement. The program
'sorts an array in ascending order.

DECLARE SUB SortArray (A%())
Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
  READ A%(Cnt)
NEXT

CALL SortArray(A%())
CLS : PRINT
PRINT "The sorted array:"

FOR Cnt = 1 TO Max
  PRINT A%(Cnt);
NEXT

DATA 12,23,789,90,545,22,1,87,43,53,52,333,24,67,88
```

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                    <Untitled>:SortArray
SUB SortArray (A%())
  FOR C1 = LBOUND(A%) TO UBOUND(A%) - 1
     FOR C2 = C1 + 1 TO UBOUND(A%)
        IF A%(C1) > A%(C2) THEN
           Temp = A%(C2)
           A%(C2) = A%(C1)
           A%(C1) = Temp
        END IF
     NEXT C2
  NEXT C1

END SUB

                         Immediate

Main: <Untitled>     Context: Program not running          00010:010
```

٢ – نفذ البرنامج. لاحظ استخدام عبارة SUB فى البرنامج.

```
The sorted array:
 1  12  22  23  24  43  52  53  67  87  88  98  333  545  789





Press any key to continue
```

٣ – احفظ هذا البرنامج كملف نص تحت اسم SUB.BAS مع اخلاء الشاشة.

٤ – انتقل الى الدرس الثانى والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس والأربعون

عبارة SWAP

الوصف

تبادل عبارة SWAP محتويات متغيرى مؤشرات، وتكوينها هو كما يلى :

```
SWAP var1, var2
```

فإذا كانت var1 = 10 و var 2 = 20 فبعد تنفيذ SWAP تصبح var 1 = 20 و var 2 = 10 ، ويمكن ان يكون المتغيران من اى نوع مسموح به فى بيسك السريع، ويجب ان يكون المتغيران من نفس النوع فإذا لم يحدث ذلك فتظهر رسالة خطأ تحدد ان النوع غير متوافق.

التطبيقات

عبارة SWAP هى وسيلة مفيده عندما يرغب المبرمج فى ابدال قيمتى متغيرين مثال ذلك عند اجراء عملية ترتيب، وفيما يلى بعض الامثلة.

مثال١

```
SWAP Pct1%,Pct2%
```

مثال٢

```
PRINT LastName$,FirstName
$SWAP FirstName$,LastName$
PRINT LastName$,FirstName$
```

مثال٣

```
TYPE  DriverRec
   DriverName AS STRING • 30
   DriverID   AS INTEGER
END TYPE
DIM DriverList(20) AS DriverRec, DriverOne AS DriverRec
..
..
SWAP DriverList(10).DriverName,DriverList(1).DriverName
```

عملية تقليدية

العملية التالية توضح استخدام عبارة SWAP. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This program uses the SWAP statement to sort an array

CLS


Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
  READ A%(Cnt)
NEXT

CLS : PRINT
PRINT "The array before sorting:"

FOR Cnt = 1 TO Max
  PRINT A%(Cnt);
NEXT

PRINT : PRINT

PRINT "The sorted array:"
GOSUB SortArray

FOR Cnt = 1 TO Max
  PRINT A%(Cnt);
NEXT
END

SortArray:
FOR C1 = LBOUND(A%) TO UBOUND(A%) - 1
  FOR C2 = C1 + 1 TO UBOUND(A%)
     IF A%(C1) > A%(C2) THEN
       SWAP A%(C2), A%(C1)
     END IF
  NEXT C2
NEXT C1

RETURN


DATA 12,23,789,90,545,22,1,87,43,53,52,333,24,67,88

                          Immediate
Main: <Untitled>     Context: Program not running          00054:003
```

٢ - نفذ البرنامج ولاحظ مخرجاته واستخدام عبارة SWAP فى البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

```
The array before sorting:
 12  23  789  90  545  22  1  87  43  53  52  333  24  67  88

The sorted array:
 1  12  22  23  24  43  52  53  67  87  88  90  333  545  789









Press any key to continue
```

٣ - من قائمة File اختر Save واكتب SWAP.BAS كاسم للملف ثم حدد أن تشكيل الملف نصى واحفظ هذا البرنامج.

٤ - من قائمة File اختر New مع اخلاء الشاشة.

٥ - انتقل الى الدرس المائة والحادى والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والسادس والأربعون

عبارة SYSTEM

الوصف

تغلق عبارة SYSTEM كل الملفات وتنهى البرنامج وتعود الى نظام التشغيل. وتكوينها هو كما يلى :

```
SYSTEM
```

وعندما تنفذ فى برنامج قائم بذاته فيخرج البرنامج الى DOS، وعندما تنفذ من بيئة تطوير بيسك السريع فيتوقف البرنامج. وعندما تنفذ من نافذة فورية Immediate window فيتم الخروج من بيسك السريع.

التطبيقات

تعامل عبارة SYSTEM متماثلة مع عبارة END فى بيسك السريع. وتتسبب عبارة SYS-TEM فى إغلاق كل الملفات وانهاء البرنامج. واختيار SYSTEM او END يرجع الى المبرمج نفسه. وفيما يلى بعض الامثلة :

مثال١

```
SYSTEM
```

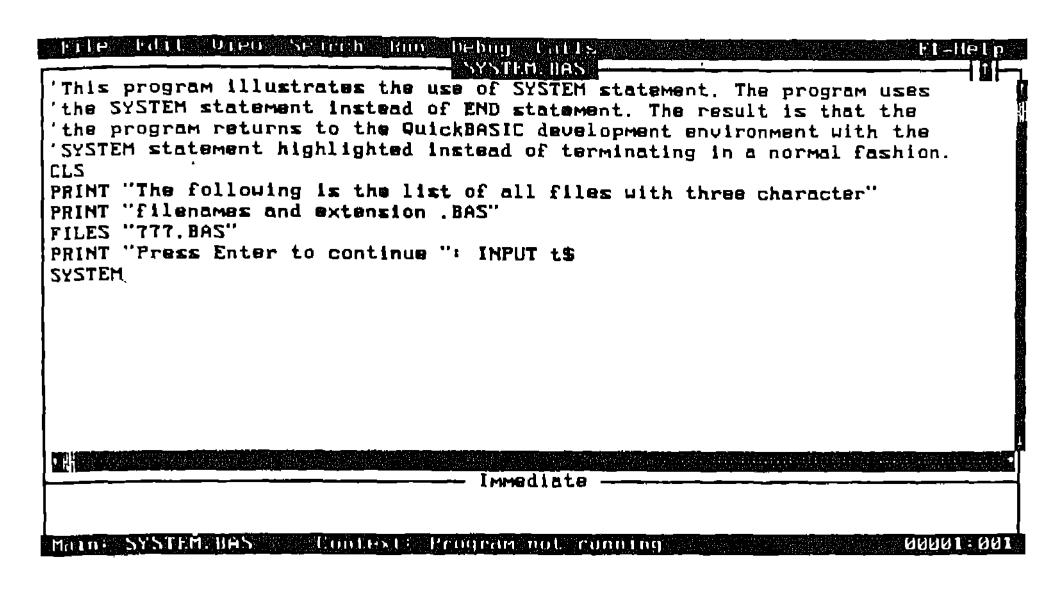
مثال٢

```
OPEN "Temp.Txt" FOR APPEND AS #10
..
..
CLS : PRINT "Goodbye .."
SYSTEM
```

عملية تقليدية

هذه العملية توضح استخدام عبارة SYSTEM، ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



٢ – نفذ البرنامج. لاحظ استخدام عبارة SYSTEM فى البرنامج. وتبين الشاشة التالية شاشة بيسك السريع بعد انتهاء البرنامج. لاحظ انها لا تكون مختلفة .

```
The following is the list of all files with three character
filenames and extension .BAS
C:\QB
BOX     .BAS      ASC     .BAS      ABS     .BAS      SUB     .BAS
STR     .BAS      A       .BAS
 2070528 Bytes free
Press Enter to continue
?
```

٣ – ارجع الى البرنامج واختر New ولا تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والثانى والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع والأربعون

دالة TAB

الوصف

تتسبب دالة TAB فى ترحيل مخرجات البرنامج عددا محددا من الفراغات ناحية اليمين، وتستخدم TAB مع عبارات PRINT و LPRINT فقط. وتكوينها هو كما يلى :

```
TAB(numeric expression)
```

يجب ان ينتج عن التعبير العددى رقما صحيحا (يقع من المدى 0 الى 32,767) . وينتقل وضع الطباعة الى العمود المحدد فى التعبير العددى. فإذا كان العمود ابعد من عرض السطر الحالى فينتقل وضع الطباعة الى السطر التالى .

التطبيقات

دالة TAB مفيدة فى التحكم فى وضع الطباعة بدقة مرتفعة. ويمكن استخدامها فى طباعة تقارير وفى التغذية المرتجعة للبرنامج، وفيما يلى بعض الامثلة :

```
PRINT TAB(10) "Name :" TAB(3) LastName$ "," FirstName$
PRINT TAB(228) "Here you are"
PRINT TAB(1209) "Here you are again !"
LPRINT "File Name:" TAB(10) FileName$
```

عملية تقليدية

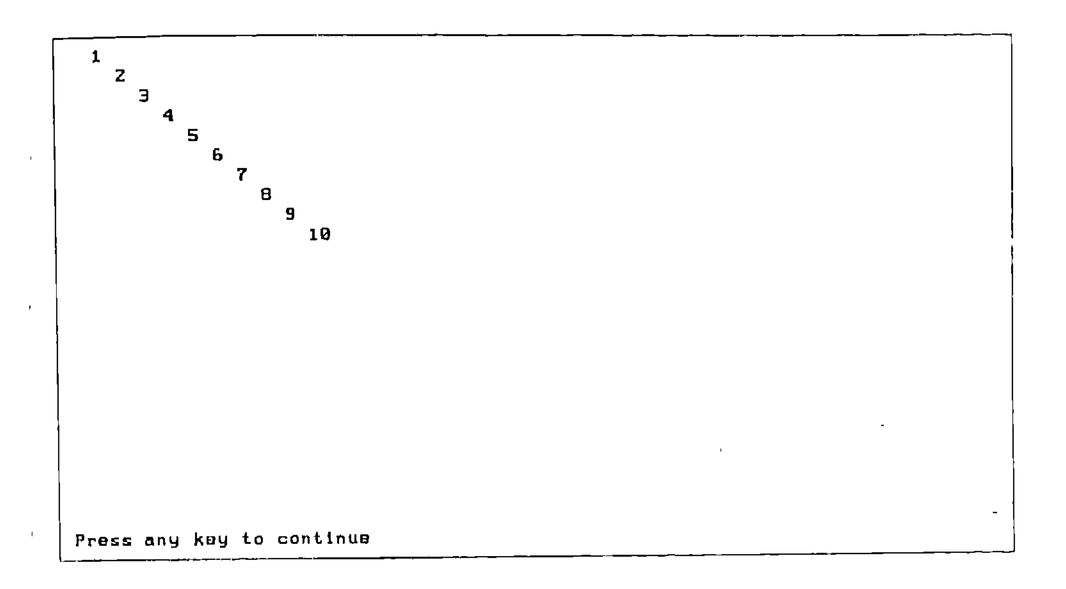
المثال التالى يوضح استخدام دالة TAB. ابدأ بتحميل بيسك السريع .

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                          <Untitled>
FOR Cnt = 1 TO 18
   PRINT TAB(Cnt * 2); Cnt
NEXT
```

```
Immediate
Main: <Untitled>          Context: Program not running          00001:005
```

٢ - اضغط على Shift - F5 لتنفيذ البرنامج. لاحظ استخدام دالة TAB لتعبير عددى فى تحديد وضع طباعة قيمة عداد FOR .



٣ - اضغط على اى مفتاح للعودة الى قائمة البرنامج. اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لاخلاء الشاشة.

٤ - انتقل الى الدرس المائة والرابع والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثامن والأربعون

دالة TAN

الوصف

تعطى دالة TAN قيمة ظل الزاوية معبرا عنها بالتقدير الدائرى . وتكوينها هو كما يلى :

```
TAN(numeric expression)
```

تحسب قيمة TAN بدقة فردية كقيمة تقليدية . وعندما يكون التعبير العددى له دقة مزدوجة فتحسب قيمة TAN بدقة مزدوجة كذلك. والتعبير العددى هو الزاوية بالتقدير الدائرى . وعندما يحدث سريان زائد لدالة TAN فيتوقف البرنامج إلا إذا كان هناك خطأ فى البرنامج.

التطبيقات

تستخدم دالة TAN عندما يراد حساب ظل الزاوية كما فى حالة الرسومات والتطبيقات الرياضية. وفيما يلى بعض الامثلة.

```
Elevation = Base • TAN(Angle)

PRINT "Simulated TAN result "; STan;
PRINT "TAN function result "; TAN(q)
```

عملية تقليدية

العملية التالية توضح استخدام دالة TEN . ابدأ بتحميل بيسك السريع.

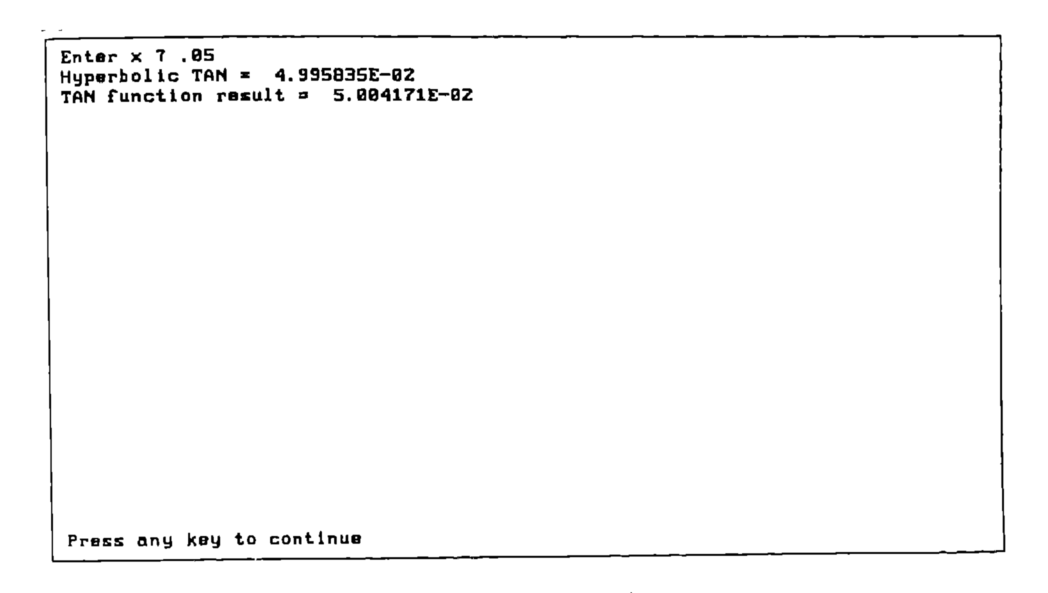
١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                F1=Help
                          <Untitled>
'This program prints the hyperbolic TAN and the result of the
'TAN function. tanH x = e^x - e^-x / e^x + e^-x
CLS
INPUT "Enter x "; x
y = EXP(x) - EXP(-x)
z = EXP(x) + EXP(-x)
PRINT "Hyperbolic TAN = "; y / z
PRINT "TAN function result = "; TAN(x)
```

Immediate

Main: <Untitled>   Context: Program not running

٢ – نفذ البرنامج. اكتب 05 على انها x (تذكر ان هذه الدالة تكون بالتقدير الدائرى) ولاحظ استخدام دالة TAN فى البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.



٣ – من قائمة File اختر New لإخلاء الشاشة.

٤ – انتقل الى الدرس المائة والثامن والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع والأربعون

دالة وعبارة TIME$

الوصف

يمكن ان تستخدم الكلمة الرئيسية TIME$` كدالة او عبارة. وكعبارة تضبط TIME$ ساعة الكمبيوتر لوقت محدد بالنسبة لليوم. وكدالة تعود TIME$ بالوقت الحالى طبقا لساعة الكمبيوتر. وتكوين كل منهما هو على النحو التالى :

تكوين عبارة TIME$:

```
TIME$=String expression
```

الجزء	الوصف
TIME$	كلمة من كلمات بيسك السريع المحجوزة.
string expression	الساعة والدقيقة والثانية مفصولة عن بعضها البعض وموضوعة بين علامتى تنصيص مزدوجتين. ويعبر عن الوقت على اساس ساعة بها 24 ساعة بدلا من ان تكون بها 12 ساعة

تكوين دالة TIME$:

```
TIME$
```

يعيد هذا التكوين الوقت الحالى طبقا لساعة الكمبيوتر الداخلية. والوقت الذى تعيده مبنى على ساعة بها 24 ساعة فى شكل hh:mm:ss حيث hh هى الساعات و mm هى الدقائق و ss هى الثوانى .

التطبيقات

تستخدم عبارة او دالة TIME$ عندما يحتاج البرنامج ان يتصل بساعة الكمبيوتر. وفيما يلى امثلة لنوعى الاستخدام.

تحديد الوقت باستخدام عبارة TIME$:

```
TIME$ = "08:30:00"        ' set the clock to 8:30 A.M.
TIME$ = "20:30:00"        ' set the clock to 8:30 P.M.
TIME$ = "07"              ' set the clock to 7:00 A.M.
TIME$ = "10:27"           ' 10:27 A.M.
TIME$ = "00:00:00"        ' initialize to zero
```

الحصول على الوقت باستخدام دالة TIME$:

```
T$ = TIME$                ' Get the time into T$
PRINT TIME$               ' print the time
T$ = TIME$
Hour = Val(T$)
IF Hour >= 12 THEN D$ = "PM" ELSE D$ = "AM"
PRINT T$ " " D$
                          ' Print the time and if it is AM or PM
```

عملية تقليدية

العملية التالية تستخدم دالة TIME$ فى توضيح بسيط. ابدأ بتحميل بيسك السريع.

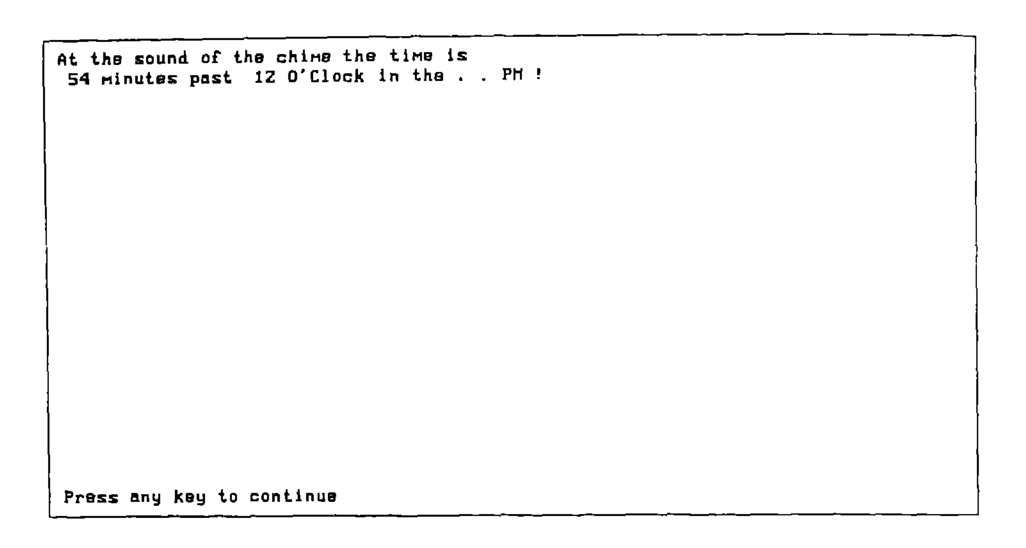
١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                         <Untitled>
'This program demonstrates the TIME$ command
T$ = TIME$
Hr% = VAL(TIME$)
M$ = MID$(T$, 4, 2)
Min% = VAL(M$)
IF (Hr% >= 12) AND (Hr% <> 24) THEN
   ap$ = "in the . . PM!"
   IF Hr% <> 12 THEN Hr% = Hr% - 12
ELSE
   ap$ = "in the . . AM !"
   IF Hr% = 24 THEN Hr% = 12
END IF
CLS
SOUND 1100, 8
PRINT "At the sound of the chime the time is "
PRINT Min%; "minutes past "; Hr%; "O'Clock "; ap$

                         Immediate

Main: <Untitled>     Context: Program not running          00016:050
```

٢ - نفذ البرنامج ولاحظ المخرجات واستخدام TIME$ فى البرنامج .



```
At the sound of the chime the time is
 54 minutes past  12 O'Clock in the . . PM !

Press any key to continue
```

٣ - اضـغط على اى مـفـتـاح للعـودة الى البـرنامج . من قـائمـة File اختـر Save واكـتب TIME.BAS كاسم للملف وحدد أن شكل الملف نصى واحفظ الملف .

٤ - من قائمة File اختر New .

٥ - انتقل الى الدرس الثامن والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والخمسون

دالة TIMER

الوصف

تعطى دالة TIMER الوقت بعدد الثوانى المنقضية من الساعة 12.00 am. وتكوينها هو كما يلى :

```
TIMER
```

تعيد الدالة عددا يتراوح بين 1 و 86,000 (86,000 هو الوقت بالثوانى لعدد 24 ساعة).

التطبيقات

عادة ما تستخدم دالة TIMER مع عبارة RANDOMIZE ، وهناك استخدامات عديدة أخرى تتحدد طبقا للتطبيق فقط. وفيما يلى امثلة لاستخدام دالة TIMER .

مثال١

```
RANDOMIZE TIMER
..
```

مثال٢

```
Start = TIMER
FOR R = 1 TO 1000
      ..
NEXT
PRINT (Start-(TIMER)); "Seconds"
```

يستخدم هذا المثال دالة TIMER فى ايجاد طول الفترة التى يستغرقها تنفيذ دورة FOR .. NEXT .

عملية تقليدية

هذه العملية تسخدم دالة TIMER فى وضع قيمة ابتدائية لمنتج ارقام عشوائية . وقد تم انتاج البرنامج فى هذه العملية فى الدرس المائة والثالث عشر. ابدأ بتحميل بيسك السريع.

١ – اخـتـر Open وحـمل البـرنامج الذى سـبق انتـاجـه فى الدرس المائة والثـالث عـشـر،

RANDOM.BAS

٢ – نفذ البرنامج، لاحظ استخدام دالة TIMER لوضع الاساس لمنتج الارقام العشوائية . وهذه طريقة جيدة لوضع اساس منتج الارقام العشوائى لان نتيجة TIMER تكون مختلفة دائما وهذا لضمان ان الاساس يكون فريدا فى كل مرة.

٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والحادى والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والحادى والخمسون

عبارات TIMER ON و TIMER OFF و TIMER STOP

الوصف

عبارة TIMER ON : عبارة تمكن من اصطياد الاحداث لدالة TIMER. والاحداث التى تحدث بعد تنفيذ هذه العبارة يمكن اصطيادها هى والحدث الذى يحدث طبقا لها .

عبارة TIMER OFF : عبارة تلغى من إمكانية اصطياد الاحداث . والاحداث التى تحدث بعد تنفيذ هذه العبارة تهمل ولا يمكن تذكرتها لاستخدامها عندما تنفذ عبارة TIMER ON بعد ذلك.

عبارة TIMER STOP : عبارة توقف اصطياد الاحداث. والاحداث التى تحدث بعد تنفيذ هذه العبارة يتم تذكرتها واصطيادها بعد تنفيذ عبارة TIMER ON تالية.

التطبيقات

تستخدم عبارات TIMER ON و TIMER OFF و TIMER STOP فى اغراض TIMER لاصطياد الاحداث . وتعتمد الاستخدامات على التطبيق نفسه. وفيما يلى بعض امثلة لعبارات TIMER ON و TIMER OFF و TIMER STOP .

```
TIMER ON
ON TIMER(120) GOSUB Process1
..
ON TIMER(1200) GOSUB Process2
..
Process1:
     ..
     RETURN
Process2:
     ..
     TIMER OFF
     RETURN
```

يبين المثال السابق استخدام TIMER OFF فى انهاء اصطياد احداث TIMER للبرنامج.

عملية تقليدية

هذه العملية تستخدم البرنامج الذى سبق اعداده فى الدرس الثانى والتسعين. ويستخدم البرنامج عبارة TIMER ON فى تتبع الوقت المنقضى بين ظهور ملقن المدخلات والاستجابة له.

ابدأ بتحميل بيسك السريع.

١ - اختر فتح OPEN وحمل البرنامج الذى سبق اعداده فى الدرس الثانى والتسعين ONEVENT.BAS . عدل البرنامج كما هو مبين فى قائمة البرنامج التالية :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                          ONEVENT.BAS
'This program demonstrates the use of the ON KEY(n) GOSUB statement
'and the TIMER ON statement.
'Turn the trapping on for function key F1. Trap the event and
'branch to GetOut. Branch from GetOut to Finit, and end the program.
TIMER ON
KEY(1) ON
ON KEY(1) GOSUB GetOut
ON TIMER(2) GOSUB RingBell
CLS

DO
   InCh$ = INKEY$

LOOP UNTIL True

Finit:
  END

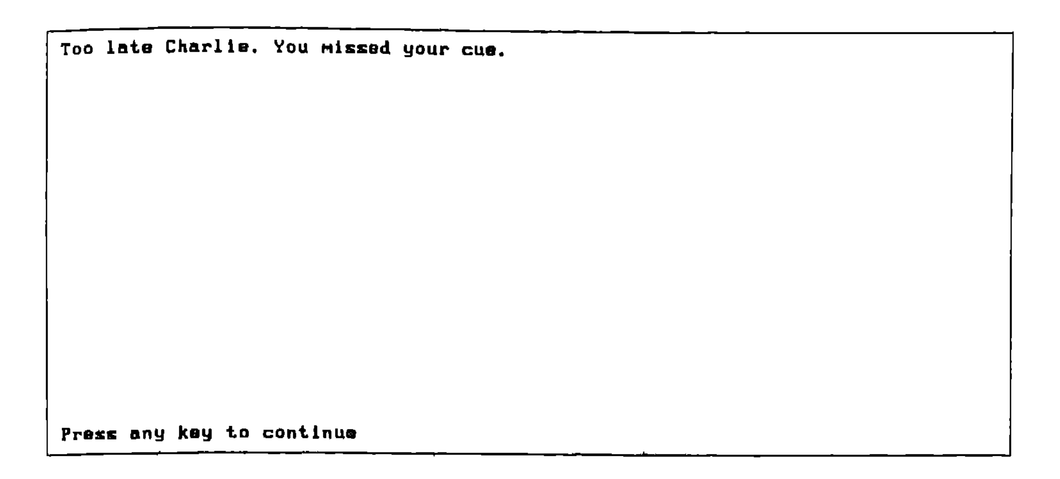
GetOut:
  PRINT "You did it! You finally found out how to stop this program."
  BEEP: BEEP
  RETURN Finit

RingBell:
  BEEP: BEEP
  PRINT "Too late Charlie. You missed your cue."
  END

                          Immediate

Main: ONEVENT.BAS     Context: Program not running
```

٢ - نفذ البرنامج. لاحظ انك اذا ما انتظرت لمدة ثانيتين قبل الضغط على F1 فيقوم البرنامج بتصيد هذا الحدث وينفذ GOSUB مع علامة الملاحظة . وفيما يلى عينة للتنفيذ والمخرجات.



٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس العشرين للاستمرار في تسلسل التعلم.

الدرس المائة والثانى والخمسون

عبارتا TRON و TROFF

الوصف

عبارتا TRON و TROFF تمكن من وتلغى المقدرة على تتبع تنفيذ البرنامج. وتكوينهما هو كما يلى :

```
TRON
TROFF
```

تمكن عبارة TRON من تتبع تنفيذ البرنامج. وتلغى عبارة TROFF المقدرة على تتبع تنفيذ البرنامج. واختيار Trace On من قائمة Debug فى بيسك السريع هو نفسه مثل ادخال عبارة TRON فى البرنامج. وتشتد اضاءة كل عبارة مع تنفيذها . وعندما تنفذ فى برنامج قائم بذاته فتعرض ارقام الاسطر فقط.

التطبيقات

تتكرر عبارات TRON و TROFF فى بيئة تطوير بيسك السريع (الصيغة 4.0) ويمكن تحقيق نفس التأثير من خلال قائمة Dabug . وفيما يلى مثال لذلك :

```
TRON
PRINT: PRINT "Trace on / Trace off Demonstration"
..
TROFF
```

عملية تقليدية

هذه العملية توضح استخدام عبارات TRON و TROFF . ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
'This program demonstrates the use of TRON and TROFF statements.
TRON
TYPE Customer
   CName AS STRING * 40
   Address AS STRING * 40
   LastInvoice AS STRING * 10
   Outstanding AS DOUBLE
END TYPE

DIM CreditPeriod AS INTEGER, Interest AS SINGLE
DIM NextCust AS Customer
CLS
PRINT "Credit Period needs "; LEN(CreditPeriod); " bytes."
PRINT "Interest needs "; LEN(Interest); " bytes."
TROFF
PRINT "Customer record needs "; LEN(NextCust); " bytes."
PRINT "Press Enter to continue."; INPUT t$

                              Immediate

Main: <Untitled>      Context: Program not running          00017:019
```

٢ – نفذ البرنامج. اضغط على مفتاح الادخال لانهاء تنفيذ البرنامج . لاحظ استخدام عبارات TRON و TROFF فى البرنامج. لاحظ كذلك كيف تشتد اضاءة عبارة PRINT مع تنفيذ البرنامج.

```
Credit Period needs  2  bytes.
Interest needs  4  bytes.
Customer record needs  98  bytes.
Press Enter to continue.
?
```

```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              <Untitled>
'This program demonstrates the use of TRON and TROFF statements.
TRON
TYPE Customer
   CName AS STRING * 40
   Address AS STRING * 40
   LastInvoice AS STRING * 10
   Outstanding AS DOUBLE
END TYPE

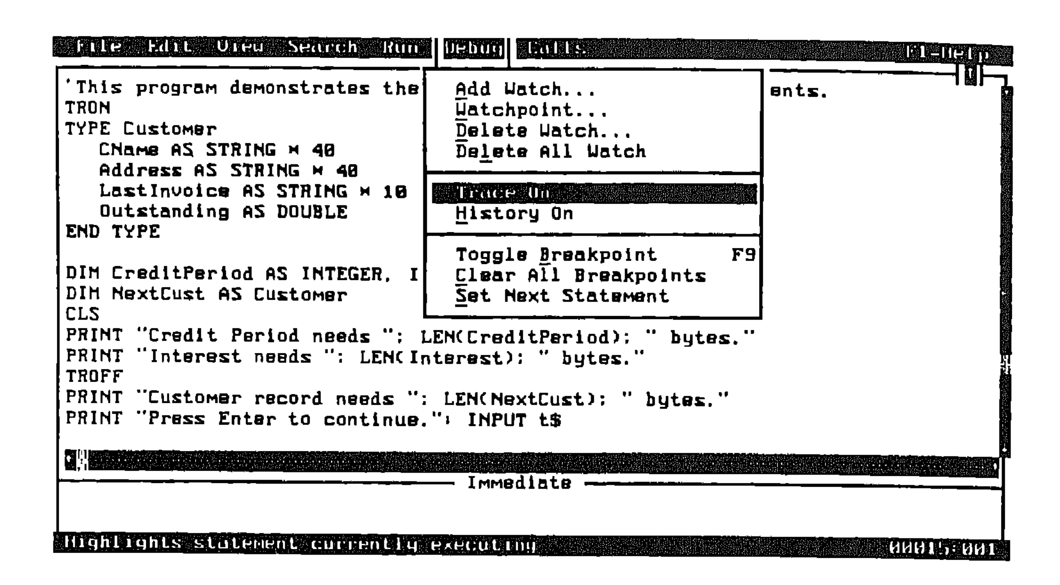
DIM CreditPeriod AS INTEGER, Interest AS SINGLE
DIM NextCust AS Customer
CLS
PRINT "Credit Period needs "; LEN(CreditPeriod); " bytes."
PRINT "Interest needs "; LEN(Interest); " bytes."
TROFF
PRINT "Customer record needs "; LEN(NextCust); " bytes."
PRINT "Press Enter to continue."; INPUT t$

                              Immediate

Main: <Untitled>      Context: <Untitled>                   00013:059
```

٣ – اضغط على اى مفتاح للاستمرار .

٤ – الشاشة التالية تبين قائمة Debug التى تقدم طريقة اخرى لتتبع البرنامج. خيار Trace فى قائمة Debug تشتد اضاءته . لرؤية ذلك . اختر قائمة Debug من المنقح.



```
 File  Edit  View  Search  Run  Debug  Calls                              F1=Help
'This program demonstrates the   Add Watch...                      ents.
TRON                             Watchpoint...
TYPE Customer                    Delete Watch...
   CName AS STRING * 40          Delete All Watch
   Address AS STRING * 40       ---------------------------
   LastInvoice AS STRING * 10    Trace On
   Outstanding AS DOUBLE         History On
END TYPE                        ---------------------------
                                 Toggle Breakpoint       F9
DIM CreditPeriod AS INTEGER, I   Clear All Breakpoints
DIM NextCust AS Customer         Set Next Statement
CLS
PRINT "Credit Period needs "; LEN(CreditPeriod); " bytes."
PRINT "Interest needs "; LEN(Interest); " bytes."
TROFF
PRINT "Customer record needs "; LEN(NextCust); " bytes."
PRINT "Press Enter to continue."; INPUT t$

------------------------------ Immediate ------------------------------

Highlights statement currently executing                          00015:001
```

٥ – ارجع الى البرنامج بالضغط على esc واختر New ولا تحفظ هذا البرنامج.

٦ – انتقل الى الدرس الثالث والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والخمسون

عبارتا TYP و END.. TYPE.

الوصف

تستخدم عبارة TYPE فى وصف متغيرات يعرفها المستفيد وتكون على هيئة مجموعة مكونة من عنصر واحد او اكثر. وتكوين العبارة ياخذ الشكل التالى :

```
TYPE
     item1 AS Type
     item2 AS Type
     ..
END TYPE
```

تسمح عبارات TYPE و END..TYPE بتجميع العناصر. والعناصر تكون مجمعة فى مجموعة وتكون اسماء متغيرات بيسك سريع صحيحة . وجزء type يمكن ان يكون من اى نوع من انواع البيانات الاساسية او من اى نوع من الانواع التى يحددها المستفيد.

عناصر السلاسل يكون طولها ثابتاً ويتم وضعها كما يلى :

```
Item1 AS STRING • length
```

جزء length يكون ثابتاً عددياً. (ولا يسمح بالتعبيرات العددية) . ولا يمكن للعناصر ان تكون اسماء منظومات. ويجب تعريف النوع الذى يحدده المستفيد قبل امكانية استخدامه.

التطبيقات

يمكن استخدام عبارة TYPE فى اى مكان فى البرنامج لوصف متغير يعرفه المستفيد. وتكون عبارة TYPE مفيدة لصيغة خاصة فى تعريف متغيرات السجل لاستخدامها مع ملفات وذلك لان البرنامج يمكن كتابته بسهولة اكبر عن كتابته باستخدام عبارة FIELD . وبالنسبة لجعل النوع الذى يعرفه المستفيد متاحا فى البرنامج فيجب ان يوصف النوع الجديد اولا ثم توضع بعد ذلك المتغيرات بانها من هذا النوع. وفيما يلى بعض امثلة لعبارة TYPE :

```
TYPE NewCust
  CustName AS STRING • 30
  CustCode AS STRING • 12

END TYPE
DIM Customer AS NewCust
```

لا يمكن استخدام النوع الذى يعرفه المستفيد NewCust فى البرنامج قبل ان تنفذ عبارة DIM .

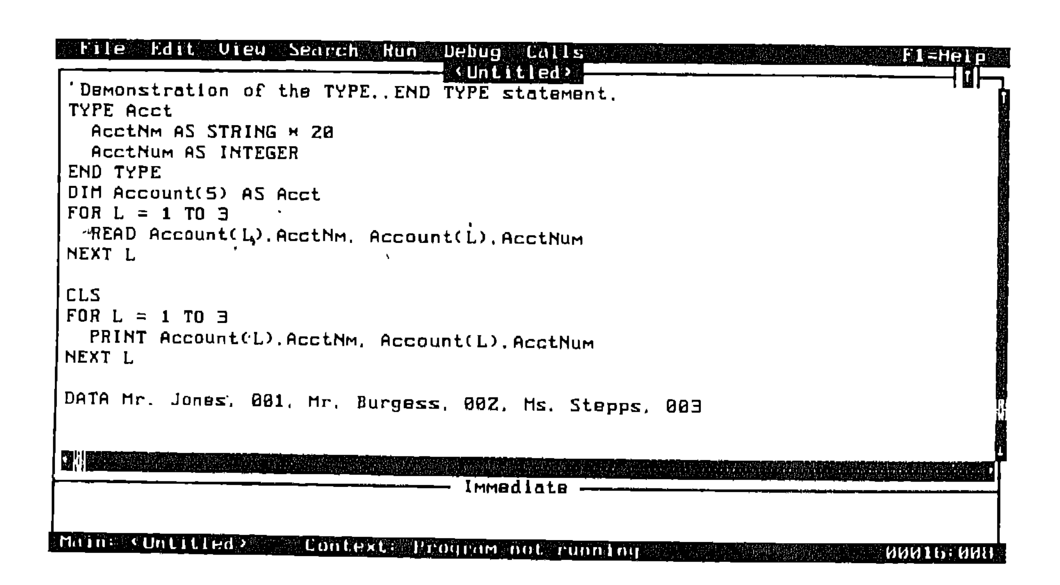
```
READ Customer.CustName, Customer.CustCode
..
TYPE Menu
  MenuItem AS STRING * 35
  Price    AS Single
END TYPE
DIM MenuList(50) AS Menu
```

عملية تقليدية

هذه العملية توضح عبارة TYPE و END.. TYPE فى برنامج بسيط . ابدأ بتحميل بيسك السريع. (ارجع الى البدء فى الدرس الثالث وملحق B).

١ – اضغط على Alt - F واكتب N لاختيار New .

٢ – اكتب البرنامج التالى :



٣ - اضـغط على Shift - F5 لتنفيذ البرنامج . لاحظ استخدام عبارة TYPE و END TYPE في البرنامج .

```
Mr. Jones                  1
Mr. Burgess                2
Ms. Stepps                 3






Press any key to continue
```

٤ - اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة .

٥ - انتقل الى الدرس الخامس والسبعين للاستمرار في تسلسل التعلم.

الدرس المائة والرابع والخمسون

دالـة VAL

الوصف

تحول دالة VAL تعبير سلسلة يحتوى على ارقام الى قيمته العددية. وتكوينها هو كما يلى:

```
VAL(string expression)
```

تبدأ دالة VAL التحـويل من الرمز الموجود على اقصى اليسار فى السلسلة وتستمر فى التحويل حتى يظهر الرمز الذى لا يكون رقما او حتى تنتهى السلسلة . ويحذف رمز عودة العربة والجداول والفراغات السابقة اثناء عملية التحويل.

التطبيقات

حيث ان تعبيرات السلاسل للارقام لا يمكن استخدامها فى العمليات الحسابية فتستخدم دالة VAL فى استخلاص القيمة العددية من تعبيرات سلاسل وتستخدم هذه القيم فى الحسابات. وفيما يلى بعض الامثلة :

```
PRINT VAL("9.22")

St$ = "111 Louis Ave."          'Address string
Blk% = VAL(St$)                 'use VAL function to extract block number
PRINT "Block " Blk%             'print block number
```

المخرجات : Block 111

عملية تقليدية

توضح العملية التالية استخدام دالة VAL. ابدأ بتحميل بيسك السريع

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the use of the VAL function.

TYPE OneType
  NName AS STRING * 20
  Address AS STRING * 30
  Phone AS STRING * 11

END TYPE

DIM Rec AS OneType

CLS
```

```
DO WHILE UCASE$(Choice$) <> "Y"
  INPUT "Enter name :"; Rec.NName
  INPUT "Address    :"; Rec.Address
  INPUT "Phone      :"; Rec.Phone

  IF VAL(Rec.Phone) = 214 THEN
    PRINT "Sounds like a local area number"
  END IF

  INPUT "Quit ? (Y/N) "; Choice$
LOOP

END
                              Immediate
```

٢ – نفذ البرنامج. اكتب Herman Munster واضغط على مفتاح الادخال اكتب 1313 Mock-ingbird Ln. واضغط على مفتاح الادخال . واخيرا اكتب214-555- 6789 واضغط على مفتاح الادخال .

```
Enter name :? Herman Munster
Address    :? 1313 Mockingbird Ln.
Phone      :? 214-555-6789
Sounds like a local area number
Quit ? (Y/N) ? y

Press any key to continue
```

٣ – لاحظ المخرجات واستخدم دالة VAL فى البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

٤ – من قائمة File اختر New لإخلاء الشاشة .

٥ – انتقل الى الدرس المائة والخامس والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس والخمسون

المتغيرات Variables

الوصف

المتغير variable هو دليل يمكن ان تتغير قيمته اثناء تنفيذ البرنامج. ويمكن ان يكون المتغير في بيسك السريع لاى نوع من انواع البيانات الاساسية التى سبق تقديمها فى الدرس الثالث والعشرين او لانواع البيانات التى يعرفها المستفيد. ويستخدم اسم المتغير variable name فى الاشارة الى إحدى القيم. وللاسماء الخواص التالية :

- يمكن ان يتكون الاسم من خانة واحدة وحتى 40 خانة كحد اقصى .
- يجب ان يبدأ الاسم باحد الحروف الهجائية (من A الى Z او من a الى z).
- لا يمكن ان يحتوى الاسم على رموز غير الحروف الهجائية والارقام والنقطة ورموز توضيح النوع : $ أو % او # او & او !.
- لا يمكن ان يبدأ الاسم بـ "FN" الا اذا كان استدعاءً لدالة.
- لا يمكن ان يكون الاسم كلمة من كلمات البيسك المحجوزة بالرغم من امكانية ادخال الكلمات المحجوزة داخله. وفيما يلى بعض الامثلة.

```
ProcRead
NewFile
SegWrite
ScreenPrint
```

توضيح المتغيرات : توضح المتغيرات من النوع الاساسى للبيانات (غير التى يعرفها المستفيد) بإضافة رمز لتوضيح النوع كلاحقة له . ورموز توضيح النوع المختلفة والانواع التى تحددها للمتغيرات هى ما يلى :

اللاحقة	النوع
$	سلسلة
%	عددى صحيح

عددى صحيح طويل	&
دقة فردية	!
دقة مزدوجة	#

امثلة :

```
Radius = 10: Radius2% = 13
MaxAmt% = 30000
Prompt$ = "Enter selection (1. .4)"
Help$ = "Use the cursor keys to choose and press ENTER to select"
MaxAmt& = 100000
MinVal! = 2.33E-12 MaxVal! = 2.32E+12
Tolerance# = .32D-22: LoadLimit# =34525576.45555
```

وفيما يلى طرق اخرى لتوضيح المتغيرات :

- عبارة DEF type تتسبب هذه العبارة فى معاملة كل المتغيرات التى تبدأ بالحروف الموجودة فى عبارة DEF كمتغيرات من هذا النوع الاساس بدون استخدام لاحقة توضيح النوع. وفيما يلى مثال لذلك :

```
DEFINT q,r,s,t
DEFSNG l,m,n
DEFDBL p
```

- عبارة DIM .. AS وفيما يلى مثال لهذه العبارة :

```
DIM Purse AS LONG
DIM Temp AS DOUBLE, Name AS STRING
```

- عبارة COMMON .. AS وفيما يلى مثال لهذه العبارة :

```
COMMON FirstName AS STRING
COMMON Salary AS SINGLE, RatePerHr AS SINGLE
```

- عبارة REDIM .. AS وفيما يلى مثال لهذه العبارة :

```
REDIM  Purse AS SINGLE
REDIM Temp AS INTEGER, Tools AS STRING
```

- عبارة SHARED .. AS وفيما يلى مثال لهذه العبارة :

```
SHARED Department AS STRING, Floor AS INTEGER
```

- عبارة STATIC .. AS وفيما يلى مثال لهذه العبارة :

```
STATIC CubicFt AS SINGLE, SalesTax AS DOUBLE
```

ويمكن انتاج انواع بيانات يعدها المستفيد من أنواع بيانات اساسية سبق تعريفها بالطريقة التالية :

- استخدام TYPE و END..TYPE فى تعريف نوع يعرفه المستفيد وعبارة DIM .. AS لتوضيحه .

```
TYPE  BinItem
   BinID AS INTEGER
   BinContent AS STRING • 25
   BinLoc AS STRING • 5
   BinStatus AS DOUBLE
   END TYPE
DIM InventoryItem AS BinItem
```

- استخدام DIM array (Size) AS فى توضيح منظومة. ويستخدم المثال التالى تعريف الانواع السابقة.

```
DIM InventoryList (100) AS BinItem
```

التطبيقات

المتغيرات هى اكثر الوسائل استخداما فى الاشارة الى بيانات. وطريقة اختيارك لتعريف وتوضيح المتغيرات وانواعها تتحكم فى كيفية معالجة البرنامج للبيانات وتخزينها . ودرجة تعقيد تكوين البيانات المستخدمة فى البرنامج تكون مؤشرا جيدا لدرجة تعقيد البرنامج. وللمتغيرات استخدامات عديدة. وفيما يلى قلة منها :

- مخزن مؤقت للبيانات.
- تخزين بيانات اثناء اجراء الحسابات.

• اختيار المستفيد للقائمة.

• تشغيل عداد للدورات.

• قراءة بيانات من وحدات.

• كتابة بيانات فى وحدات.

عملية تقليدية

توضح هذه العملية استخدام المتغيرات عن طريق استخدام البرنامج الذى سبق تقديمه فى الدرس الثالث (BOX.BAS) يستخدم المتغير Lin% فى هذا البرنامج لحفظ تتبع موقع عنصر البيانات التالى والمراد طباعته داخل الصندوق. ويعمل كعداد لعدد الاسطر. والمتغير WStr$ هو عنصر بيانات يراد كتابته وبه معلومات يراد كتابتها . ولتعديل WStr$ واجراء تجارب عليه استمر على النحو التالى :

١ - ابدأ بيسك السريع بكتابة QB من عند ملقن DOS واضغط على مفتاح الادخال .

٢ - اضغط على Alt - F واكتب 0 لتحميل البرنامج.

٣ - اضغط على Tab للذهاب الى الدليل فى جدول حوار البرنامج Load واستخدم مفاتيح الاسهم فى اختيار البرنامج BOX.BAS .

٤ - اضغط على مفتاح الادخال لتحميل البرنامج.

٥ - انتقل نقطة البداية الى الموقع الموجود فى البرنامج الذى تتحمل فيه السلسلة "Illustrated Quick BASIC" للمتغير WSTR$ وغير السلسلة بكتابة اسمك بدلا من محتواها .

٦ - اضغط على Shift - F5 لتنفيذ البرنامج . لاحظ طباعة اسمك داخل الصندوق .

تحذير

اختصر اسمك بحيث الا يزيد طوله عن 25 خانة. فلا يطبع البرنامج سلاسل اكبر من ذلك داخل الصندوق. حاول ان تفعل ذلك بنفسك.

٧ - اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٨ – اضغط على Alt - F ثم اكتب X للخروج من بيسك السريع. اضغط على b ثم على قضيب المسافات لاختيار عدم حفظ البرنامج.

٩ – انتقل الى الدرس المائة والثالث والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والسادس والخمسون

دالتا VARPTR و VARSEG

الوصف

تستخدم الدوال VARPTR و VARSEG للحصول على الفرع للذاكرة وقطاع متغير محدد. وعندما لا يتواجد المتغير مع استخدام هذه الدوال فيتم انتاج المتغير مع عودة العنوان. وعندما يكون المتغير متغير سلسلة فتعود الدوال بعنوان اول بايت من واصف السلسلة .

دالة VARPTR : تعيد هذه الدالة الفرع للمتغير. وتكوينها هو كما يلى :

```
VARPTR(variable)
```

جزء variable يمكن ان يكون اى متغير بيسك سريع من اى نوع اساسى او يعرفه المستفيد. والفرع الذى يعود يكون داخل القطاع الحالى .

دالة VARSEG : تعيد دالة VARSEG جزء القطاع لعنوان متغير محدد. وتكوينها هو كما يلى :

```
VARSEG(variable)
```

وجزء variable هو نفسه مثل المذكور فى دالة VARPTR. والعنوان الذى يعود هو القطاع المخزن داخله المتغير.

التطبيقات

يتكرر استخدام دالتى VARPTR و VARSEG فى البرمجة مختلطة اللغات. وهناك تطبيقات اخرى تستخدم مع عبارات BLOAD و BSAVE و PEEK و POKE و CALL . وفيما يلى أمثلة لدالتى VARPTR و VARSEG :

```
DIM Window1(80,25)
..
DEF SEG = VARSEG(Window1(1,1))
BSAVE "Screen.001", VARPTR(Window1(1,1)), 4078
```

عملية تقليدية

تستخدم العملية التالية البرنامج الذى سبق اعداده فى الدرس التاسع، ويخدم البرنامج كذلك كتوضيح لدوال VARPTR و VARSEG . ابدأ بتحميل بيسك السريع.

١ - اختر open وحمل البرنامج المسمى BLSAVE.BAS .

```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                 BLSAVE.BAS
'This is a demonstration of the BLOAD and BSAVE statements.

'$DYNAMIC
DIM w(10, 10)
DEF SEG = VARSEG(w(1, 1))
FOR i = 1 TO 10
  FOR i2 = 1 TO 10
    w(i, i2) = i + i2
  NEXT i2, i

'Note the length provided is the number of elements in array w plus 1
'multiplied by 4.

BSAVE "W.Dat", VARPTR(w(1, 1)), 404
FOR i = 1 TO 10
  FOR i2 = 1 TO 10
    w(i, i2) = 0
NEXT i2, i
ERASE w
REDIM w(10, 10)

BLOAD "W.Dat", VARPTR(w(1, 1))
CLS

FOR i = 1 TO 2
  FOR i2 = 1 TO 10
    PRINT (w(i, i2))
NEXT i2, i

END

                                  Immediate

Main: BLSAVE.BAS       Context: Program not running                  00036:001
```

٢ - نفذ البرنامج. لاحظ استخدام VARPTR و VARSEG فى البرنامج. تستخدم عبارة DEF SEG دالة VARSEG للحصول على قطاع المتغير وجعله القطاع الحالى . وتستخدم عبارة BSAVE دالة VARPTR فى الحصول على فرع للمتغير وكتابة صورة الذاكرة من هذا الفرع. وتستخدم عبارة BLOAD دالة VARPTR فى اعادة تحميل بيانات فى موقع الذاكرة .

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والسابع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع والخمسون

دالة VARPTR$

الوصف

دالة VARPTR$ تعطى تمثيلا على هيئة سلسلة لفرع المتغير المؤشر. وتكوينها هو كما يلى

```
VARPTR$(Variable)
```

يمكن ان يكون المتغير من اى نوع فإذا لم يكن المتغير مستخدما بالفعل فيتم انتاجه وتعيد الدالة تمثيل السلسلة للفرع. فاذا كان المتغير منظومة فيجب ان تكون المنظومة سلسلة متغير ويجب ان تحدد لها ابعاد قبل استخدام دالة VARPTR$.

التطبيقات

تستخدم دالة VARPTR$ اساسا مع عبارات PLAY و DRAW . ومع عبارات PLAY، DRAW تستخدم VARPTR$ فى تنفيذ سلاسل جزئية . وفيما يلى مثال لدالة VARPTR$.

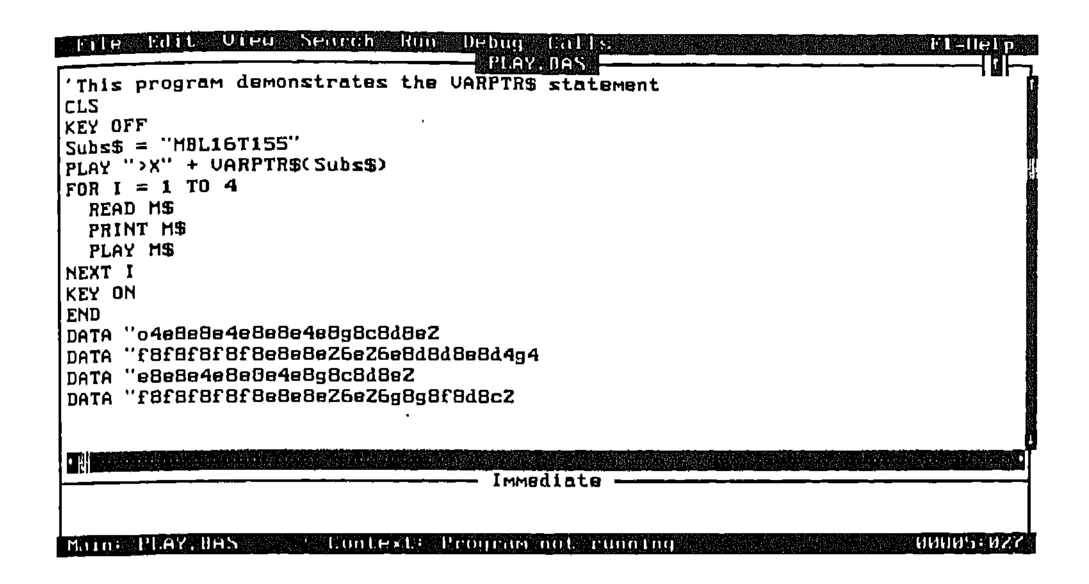
```
Subs$ = "p24 p8 18"
..
PLAY Subs$
..
PLAY ">>X" + VARPTR$(Subs$)
```

عملية تقليدية

هذه العملية توضح استخدام دالة VARPTR$ مع عبارة PLAY فى تنفيذ سلسلة جزئية .

ابدأ بتحميل بيسك السريع .

١ - اختر open وحمل البرنامج المسمى PLAY.BAS. عدل اول عبارة لتأخذ الشكل الموجود فى القائمة التالية :



```
File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                    PLAY.BAS
'This program demonstrates the VARPTR$ statement
CLS
KEY OFF
Subs$ = "MBL16T155"
PLAY ">X" + VARPTR$(Subs$)
FOR I = 1 TO 4
  READ M$
  PRINT M$
  PLAY M$
NEXT I
KEY ON
END
DATA "o4e8e8e4e8e8e4e8g8c8d8e2
DATA "f8f8f8f8f8e8e8e26e26e8d8d8e8d4g4
DATA "e8e8e4e8e8e4e8g8c8d8e2
DATA "f8f8f8f8f8e8e8e26e26g8g8f8d8c2
                                    Immediate
Main: PLAY.BAS          Context: Program not running              00005:027
```

٢ – نفذ البرنامج، لاحظ استخدام دالة VARPTR$ فى البرنامج.

٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والسابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والثامن والخمسون

عبارة VIEW

الوصف

عبارة VIEW تعرف حدود الشاشة للرسومات، وتكوينها هو كما يلى :

```
VIEW SCREEN (x1,y1)-(x2,y2),color,border
```

يحدد جزء SCREEN ان الحدود المذكورة للشاشة مطلقة وعندما لا يستخدم هذا الجزء فتكون الاحداثيات نسبية لموقع الشاشة الحالى . ويصف جزء (x1 , y1) - (x2 , y2) حدود مجال الرؤية وتكون الاحداثيات قطرية عكسية بحيث انها تكون مستطيلا . وجزء color هو خاصية اللون المستخدمة فى وصف بوابة الرؤية، ويحدد جزء border ان الحدود يجب ان ترسم حول بوابة الرؤية ، ويمكن ان تكون الحدود اى قيمة عددية، وعندما تحذف فلا ترسم حدود حول بوابة الرؤية.

وتعرف دالة VIEW منطقة مستطيلة داخل شاشة رسومات معينة بحيث يمكن ان تحتوى المخرجات داخل هذه المنطقة، وهى تشبه فصل جزء من الشاشة لاستخدامه دون التاثير على بقية الشاشة، وعند استخدام عبارة VIEW بدون قوائم فإنها تعرف محتوى الشاشة على انه مجال الرؤية ، ويجب ان تقع الاحداثيات المقدمة داخل حدود الثبات المحددة للشاشة.

التطبيقات

تستخدم عبارة VIEW فى تعريف مناطق اصغر داخل شاشات رسومات معينة لمخرجات الرسومات، وعندما تشير مخرجات الرسومات الى احداثيات خارج بوابة الرؤية الحالية فتقطع المخرجات لتناسب البوابة، وتعيد العبارات RUN و SCREEN الشاشة الى حجمها الكامل وتلغى مقدرة عبارات VIEW السابقة ، وفيما يلى امثلة لعبارة VIEW :

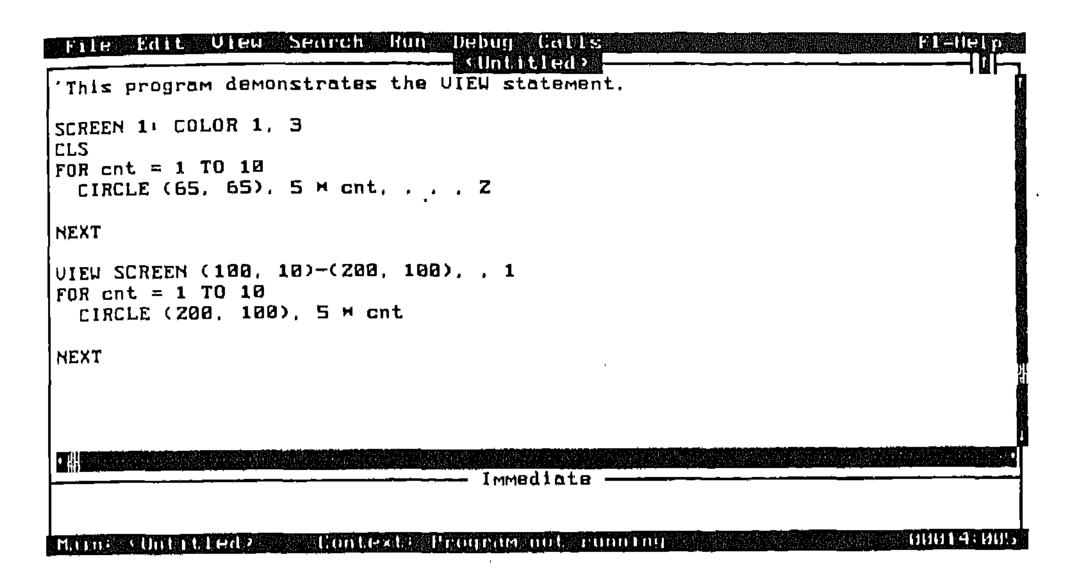
```
SCREEN 1
VIEW SCREEN (0,0)-(100,100),,1
LINE (10,10)-(50,50)

VIEW (100,10)-(200,200)
FOR C = 1 TO 20
    PSET(C,C)
NEXT
```

عملية تقليدية

هذه العمليـة توضح استـخدام عبارة VIEW فى صـورة مبـسطة، اسـتمـر إذا كانت لديك امكانيات رسومات ملونة فقط . ابدأ بتحميل بيسك السريع.

١ – اكتب البرنامج التالى :



٢ – نفذ البرنامج. لاحظ استخدام عبارة VIEW فى البرنامج.

٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والتاسع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع والخمسون

عبارة VIEW PRINT

الوصف

عبارة VIEW PRINT تعرف حدود الشاشة فى الشاشة التالية: وتكوينها هو كما يلى :

```
VIEW PRINT line1 TO line2
```

الجزمين line1 و line2 يعرفان اعلى سطر وادنى سطر فى بوابة الرؤية. وعندما يحذفان فتعرف الشاشة على انها بوابة الروية.

التطبيقات

عبارة VIEW PRINT مفيدة عندما تريد ان تعرض نصا فى جزء من الشاشة دون ان تؤثر على بقية الشاشة، والنص المعروض يدور مستقلا كذلك عن بقية الشاشة بحيث ان النصوص الموجودة فى الاجزاء الاخرى من الشاشة لاتتأثر. ولا يمكنك ان تعرف حد العمود لبوابة الرؤية. وفيما يلى امثلة لعبارة VIEW PRINT .

```
CLS
PRINT "Look at the bottom of the screen ...."
VIEW PRINT 10 TO 22
FOR Cnt = 1 TO 20
     PRINT SPC(Cnt);"Scrolling away...."
NEXT

VIEW PRINT 1 TO 5
LOCATE 3,3
PRINT "The LOCATE statement works inside the view port"
```

عملية تقليدية

هذه العملية عبارة عن توضيح بسيط لعبارة VIEW PRENT . ابدأ بتحميل بيسك السريع .

١ – اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                                <Untitled>
'This program demonstrates the use of the VIEW PRINT statement.
'The program requests a filename and lists that file.

CLS
FILES "????.BAS"
INPUT "Enter file to list: "; FileName$

IF FileName$ <> "" THEN
   OPEN FileName$ FOR INPUT AS #1
   VIEW PRINT 10 TO 24: CLS
   InCh$ = INPUT$(1, #1)
   DO WHILE NOT EOF(1)
      IF (InCh$ <> CHR$(13)) THEN PRINT InCh$;
      InCh$ = INPUT$(1, #1)
   LOOP
   CLOSE #1
ELSE
   PRINT : PRINT "Thank you for participating ..."
END IF

                               Immediate

Main: <Untitled>      Context: Program not running               00036:007
```

٢ – نفذ البرنامج واكتب LINE.BAS ولاحظ دوران القائمة . لاحظ استخدام عبارة VIEW PRINT فى البرنامج.

```
C:\QB
BOX      .BAS      BOXZ     .BAS      CASE     .BAS      ASC      .BAS
ABS      .BAS      PLAY     .BAS      OPEN     .BAS      LINE     .BAS
DRAW     .BAS      LPOS     .BAS      SUB      .BAS      STR      .BAS
A        .BAS
 2029560 Bytes free
Enter file to list: ? LINE.BAS


    t% = A%(Cnt)
    'Compute the height of the bar
    y2 = t% * 2
    'Draw the box
    LINE (x1 + 5, y1)-(x2, y2), 2, BF
    'Move both x-coordinates to the right
    x1 = x1 + 20: x2 = x2 + 20
NEXT


DATA 13,25,44,65,32,22,8,5,71,8




Press any key to continue
```

٣ – ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ – انتقل الى الدرس المائة والثانى والستين للاستمرار فى تسلسل التعلم.

الدرس المائة والستون

عبارة WHILE.. WEND

الوصف

تستخدم عبارة WHILE.. WEND فى تكرار تنفيذ سلسلة من عبارات البرنامج حتى لايتحقق شرط معين، وتكوينها هو كما يلى :

```
WHILE condition
   program statements
   ..
WEND
```

والشرط هو تعبير بوليان يقوم على أنه صحيح أو خطأ. وتنفذ عبارات البرنامج الموجود داخل مجموعة WHILE.. WEND مرات ومرات طالما أن الشرط متحقق، وعندما تصبح قيمة الشرط خطأ فيستمر التنفيذ بالعبارة التى تلى عبارة WEND، ويمكن أن تتداخل عبارات WHILE.. WEND إلى أى مستوى، وعندما لايتوافق عدد من عبارات WHILE مع عدد من عبارات WEND فينتج المترجم رسالة خطأ. فإذا تعدت عبارات WEND عبارات WHILE فتظهر رسالة خطأ بوجود WEND بدون WHILE، وإذا تعدت عبارات WHILE عبارات WEND فتظهر رسالة خطأ تدل على ذلك أيضاً.

ويجب عمل تداخل أى مكون تحكم بحذر لضمان الحفاظ على سلامة منطق البرنامج، واحدى طرق عمل ذلك هى فحص مجموعات عبارات البرنامج الموجودة داخل مكون التحكم المتداخل بعناية.

التطبيقات

عبارة WHILE.. WEND هى وسيلة مفيدة جداً فى تكرار تنفيذ مجموعة أسطر من أسطر البرنامج دون إعادة كتابتها مرات ومرات داخل البرنامج، وفيما يلى أمثلة لذلك :

مثال١

```
Q% = 0
WHILE (Q% < 10)
   GOSUB GS1
   Q% = Q% + 1
WEND
```

مثال٢

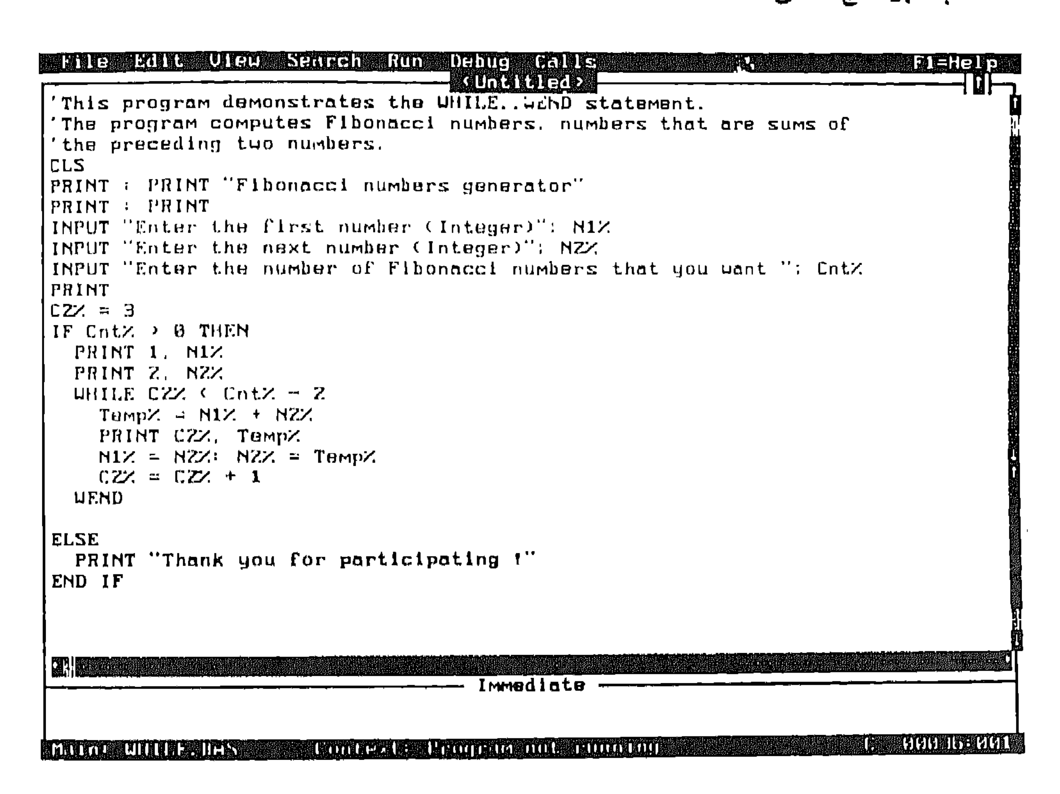
```
WHILE (Ret$ <> "Q")
   ..
   WHILE (Ret2$ <> "E')
      ..
   WEND
WEND
```

يوضح هذا المثال تداخل عبارة WHILE.. WEND.

عملية تقليدية

هذه العملية توضح استخدام عبارة WHILE.. WEND. ابدأ بتحميل بيسك السريع.

١ – أكتب البرنامج التالى :



٢ – نفذ البرنامج، لاحظ استخدام عبارة WHILE.. WEND فى البرنامج. أكتب 1 كأول رقم و 2 كالرقم الثانى و 12 كعدد لأرقام فيبوناكى الذى تريده.

```
Fibonacci numbers generator


Enter the first number (Integer)? 1
Enter the next number (Integer)? 2
Enter the number of Fibonacci numbers that you want ? 12

 1          1
 2          2
 3          3
 4          5
 5          8
 6          13
 7          21
 8          34
 9          55
 10         89
 11         144
 12         233



Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة الحادى والستون

عبارة WIDTH

الوصف

تحدد عبارة WIDTH طول السطر لأحد الملفات أو لإحدى الوحدات، وتكوينها هو كما يلى :

```
WIDTH cols, lines
WIDTH #file num|device, width
WIDTH LPRINT width
```

فى التكوين الأول جـزء cols يحدد عدد الأعمدة فى المخرجات ويحدد جزء lines عدد الأسطر فى المخرجات، والوحدة التى تتأثر هى الشاشة والقيم المسموح بها للأعمدة هى 80 و 40. والقيمة التقليدية هى 80. ويمكن أن تفترض لجزء الأسطر أى قيمة من القيم التالية : 25 أو 30 أو 43 أو 50 أو 60.

وفى التكوين الثانى يحدد القضيب الرأسى (|) إما رقم الملف filenum# أو الوحدة التى يمكن استخدامها. ورقم الملف هو الرقم المحدد للوحدة كملف مفتوح بعبارة OPEN. والوحدة هى وحدة DOS صحيحة ويعبر عنها بسلسلة مثل "LPT1". وعندما يستخدم رقم الملف فيكون أثر WIDTH فورياً وعندما تستخدم الوحدة فيبدأ تأثير WIDTH بعد عبارة OPEN التالية على الوحدة.

وفى التكوين الثالث يتأثر طابع الأسطر ويكون تأثير عبارة WIDTH فورياً. وجزء width هو عدد الأعمدة للطابع.

التطبيقات

عبارة WIDTH تكون مفيدة عندما يراد إجبار المخرجات لتأخذ عرضاً محدداً سواء كان ذلك بسبب قيود الوحدة أو طبقاً لمتطلبات التطبيق. وفيما يلى أمثلة لعبارة WIDTH.

مثال١

```
OPEN "LPT1:" FOR OUTPUT AS #1
WIDTH #1, 55
```

مثال٢

```
WIDTH LPRINT 60
```

مثال٣

```
WIDTH 25, 40
```

عملية تقليدية

هذه العملية توضح عبارة WIDTH على طابع الأسطر. استمر إذا كان لديك طابع متصل بجهاز الكمبيوتر المتاح لك فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File  Edit  View  Search  Run  Debug  Calls                          F1=Help
                              <Untitled>
'This program demonstrates the use of the LINE INPUT and LINE INPUT #
'statements. The program accepts text, writes it to a file,
'reads the text back and writes it to the screen.

CLS
PRINT "Enter text: (Enter blank line to terminate)"

OPEN "Notes.001" FOR OUTPUT AS #2

Line$ = " "
DO WHILE Line$ <> ""
   LINE INPUT ">"; Line$
   WRITE #2, Line$
LOOP
CLOSE #2

CLS
PRINT "What you typed in is as follows: "
WIDTH LPRINT 55

OPEN "Notes.001" FOR INPUT AS #2
DO WHILE NOT EOF(2)
   LINE INPUT #2, Line$
   LPRINT Line$
LOOP

CLOSE #2

                               Immediate

Main: <Untitled>     Context: Program not running                00036:009
```

٢ - نفذ البرنامج ولاحظ استخدام عبارة WIDTH والتأثير على مخرجات الطابع.

٣ - اكتب : This example shows the effect of the WIDTH LPRINT command
واضغط على مفتاح الادخال. اضغط على مفتاح الادخال مرة أخرى لإنهاء البرنامج. يظهر السطر الذى كتبته كسطرين مطبوعين.

٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٥ - انتقل إلى الدرس التاسع والتسعين للاستمرار فى تسلسل التعلم.

الدرس المائة والثاني والستون

عبارة WINDOW

الوصف

تسمح عبارة WINDOW بإعادة تعريف احداثيات الشاشة بنظام احداثيات منطقى جديد.
وتكوينها هو كما يلى :

```
WINDOW SCREEN (x1,y1)-(x2,y2)
```

جزء (x1, y1) - (x2, y2) يحدد حدود الاحداثيات الجديدة لمخرجات الرسومات (x1, y1) تمثل الركن العلوى الأيسر وتمثل (x2, y2) الركن السفلى الأيمن. ويحول جزء SCREEN الاحداثيات بحيث إن القيم تتحرك من صغير إلى كبير ومن القاعدة إلى القمة.

التطبيقات

تستخدم عبارة WINDOW عندما تحتاج مخرجات الرسومات إلى نظام احداثيات مختلف عن الاحداثيات الطبيعية لحالة معينة للشاشة. ويمكن تعريف مدى الرؤية داخل نظام الاحداثيات الجديد باستخدام عبارة VIEW. وعندما تستخدم عبارة WINDOW بدون قوائم فإنها تلغى مقدرة أى إعداد سبق اعداده بواسطة WINDOW وفيما يلى أمثلة لعبارة WINDOW.

مثال١

```
SCREEN 1
WINDOW (-50,1)-(50,50)
LINE (-50,1)-(-25,25)
..
```

مثال٢

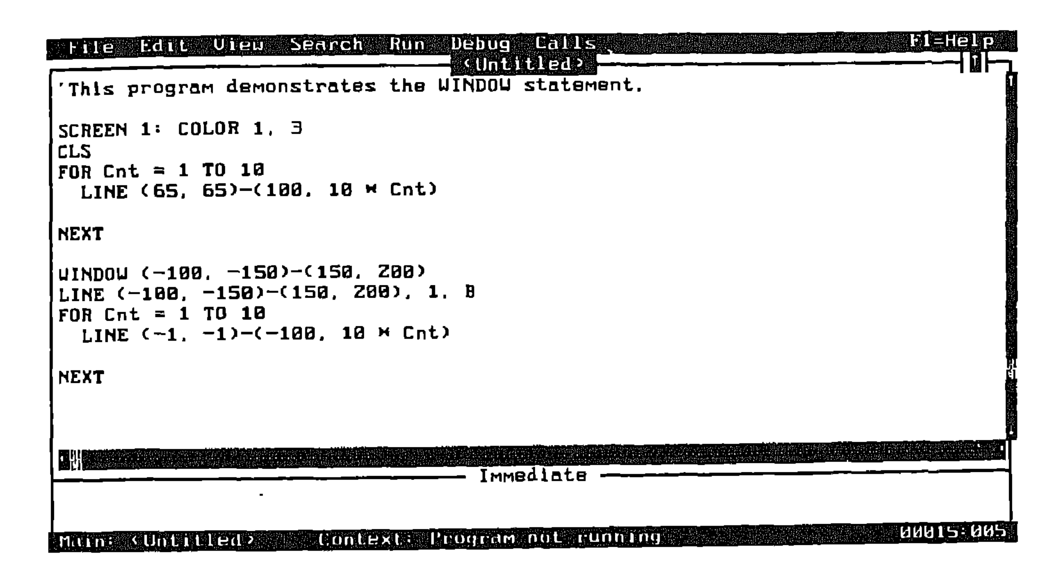
```
WINDOW SCREEN(-50,1)-(50,50)
LINE (-50,1)-(25,25)
..
```

يرسم المثال الأول سطراً من القاعدة إلى القمة ويرسم المثال الثانى سطراً من القمة إلى القاعدة. وهذا بسبب أن الاحداثيات تذهب طبيعياً من صغير فى القمة إلى كبير فى القاعدة وفى المثال الثانى يعكس جزء SCREEN ذلك.

عملية تقليدية

عبارة WINDOW موضحة فى صورة مبسطة فى هذه العملية. استمر إذا كانت لديك إمكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج. لاحظ استخدام عبارة WINDOW فى البرنامج.

٣ - ارجع إلى البرنامج واحفظ هذا البرنامج كملف نصى تحت اسم ملف WINDOW.BAS.

٤ - انتقل إلى الدرس المائة وخمسة للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والستون

عبارتا WRITE و WRITE

الوصف

عبارة WRITE : تكتب العبارة البيانات على الشاشة، وتكوينها هو كما يلى :

```
WRITE expression list
```

جزء expression list هو قائمة بعناصر البيانات المراد كتابتها على الشاشة. وتستخدم فواصل كمحددات فى هذا الجزء وتوضع سلسلة البيانات بين علامتى تنصيص مزدوجتين. تكتب البيانات العددية على الشاشة بدون فراغات سابقة لها أو تابعة لها. وتكتب عبارة WRITE تسلسل عودة العربة وتغذية السطر بعد expression list.

عبارة #WRITE : تكتب العبارة البيانات فى ملف، وتكوينها هو كما يلى :

```
WRITE #filenum, expression list
```

جزء filenum هو رقم الملف المحدد للملف فى عبارة OPEN. وجزء expression list هو قائمة بعناصر البيانات المراد كتابتها فى الملف، وتوفر سلسلة البيانات داخل علامتى تنصيص مزدوجتين وتستخدم فواصل داخل قائمة التعبير كمحددات. وتضيف عبارة #WRITE فواصل كمحددات عند الكتابة فى الملف، وتكتب البيانات العددية بدون فراغات سابقة لها أو تابعة لها، ويكتب تسلسل حركة العربة وتغذية السطر بعد expression list. وبدون قائمة التعبير تكتب عبارة #WRITE سطراً فارغاً فى الملف.

التطبيقات

عبارة WRITE : تستخدم العبارة عندما تكون هناك حاجة إلى تحكم أقل فى المخرجات وذلك لأنها لا تقدم إمكانيات تشكيل مثل عبارة PRINT. وفيما يلى بعض الأمثلة :

```
T$ = "You don't have any pie neither !"
Remainder = (100/33) - 3
WRITE 12.0,T$,Remainder
```

عبارة WRITE# : عبارة WRITE# مفيدة فى انتاج ملفات مع عناصر بيانات محددة. أحد استخدامات مثل هذا الملف هو استخدام عبارة INPUT# فى تحميل القيم داخل المتغيرات مباشرة. وفيما يلى بعض الأمثلة :

```
OPEN "Train.Log" FOR OUTPUT AS #4
..
WRITE #4 Name$, TrnType$, TrnScore
WRITE #4
```

فى هذا المثال تكتب عبارة WRITE# بدون قائمة تعبير سطراً فارغاً فى الملف.

عملية تقليدية

هذه العملية توضح عبارات WRITE و WRITE# . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
 File  Edit  View  Search  Run  Debug  Calls                              F1=Help
                                  <Untitled>
'The following program demonstrates the WRITE and WRITE # statements.
'The program generates data to write to a file and writes the same data
'to the screen using WRITE and to the disk using WRITE #.  The disk file
'is viewed using the DOS TYPE command.

CLS
PRINT "Gentlemen, behold the difference between PRINT and WRITE !"
PRINT
var1$ = "No way Jose!": var2$ = "Dem's de facts.": var3 = 248
PRINT "The PRINT statement:"
PRINT var1$, var2$, var3
PRINT "The WRITE statement:"
WRITE var1$, var2$, var3

OPEN "WP.Txt" FOR OUTPUT AS #1
WRITE #1, var1$, var2$, var3
CLOSE #1

                                  Immediate

Main: <Untitled>        Context: Program not running                00017:009
```

٢ - نفذ البرنامج. لاحظ استخدام عبارات WRITE و WRITE# فى البرنامج.

```
Gentlemen, behold the difference between PRINT and WRITE !

The PRINT statement:
No way Jose!  Dem's de facts.              248
The WRITE statement:
"No way Jose!","Dem's de facts.",248

C:\QB>type wp.txt
"No way Jose!","Dem's de facts.",248

C:\QB>
```

٣ - أخرج من بيسك السريع دون أن تحفظ البرنامج. وعند ملقن DOS اكتب WP.TXT واضغط على مفتاح الادخال. تبين الشاشة (بملقن مختلف) ما يلى :

```
C:\BASIC\QB.400\ILL.QB>type wp.txt
"No way Jose!","Dem's de facts.",248

C:\BASIC\QB.400\ILL.QB>
```

٤ - انتقل إلى الدرس الثالث والخمسين للاستمرار فى تسلسل التعلم.

ملحق A

الاصطلاحات وتعريفاتها

الوصف

فى هذا الملحق تعرف بعض مصطلحات الكمبيوتر والبرمجة المستخدمة فى هذا الكتاب. وكلمات بيسك السريع المحجوزة والتى كانت موضوع محتويات الأجزاء معرفة فى أول مقطع لكل جزء مناظر لها . وعلى هذا لم يتكرر تعريف هذه الكلمات فى هذا الملحق.

الاصطلاح	التعريف
ASCII	اختصار الشفرة النمطية الأمريكية لتبادل المعلومات American Standard Code of Information Interchange. وتستخدم كنمطية لتشكيل الرموز والتنقيط وشفرات تحكم البيانات المستخدمة بواسطة الكمبيوتر.
Boolean expression	تعبير بوليان وهو تعبير تكون نتيجته صحيح TRUE أو خطأ FALSE.
Buffer	ذاكرة احتياطية وهى آلية تسمح لشيئين أن يعملا معاً عندما يكون أحدهما أقل من الآخر.
Clipboard	لوحة القص. وفى بيسك السريع تكون لوحة القص هى المكان الذى يخزن فيه النص بعد عملية قص cut وقبل عملية لصق paste.
Control structures	تكوينات تحكم وهى آليات مقدمة فى لغات البرمجة تسمح للمبرمج بالتحكم فى منطق البرنامج وسريان البيانات. وعبارات بيسك السريع التالية : IF.. THEN.. ELSE و FOR.. NEXT و DO LOOP و WHILE.. WEND هى التى تعمل كهذه الآليات.
Data file	ملف بيانات. وهو ملف معلومات مرتب بطريقة يسبق تحديدها .
Debugging	عملية تصحيح أخطاء البرنامج.
Defaults	هذه هى قيم تستخدم عندما لا يريد المستفيد تحديد قيماً جديدة. أى إنها قيم تقليدية.

الاصطلاح	التعريف
Dialog Boxes	صناديق حوار وهى طريقة تداخل المستفيد مع نظم البرامج فى بيسك السريع. ويمكن أن يحتوى صندوق الحوار على خيارات للاختيار منها ومعلومات تقدم لنظم البرامج وتنفيذ اجراءات بواسطة نظم البرامج وتأكيد العملية وما إلى ذلك.
Directory	دليل وهو قائمة بالملفات المخزنة على القرص.
Extension	توسع يكون مكوناً من رمز واحد أو اثنين أو ثلاثة ويستخدم كلاحقة لاسم الملف. وعادة ما يستخدم فى تحديد نوع الملف.
Filename	اسم ملف وهو اسم يعطى بواسطة المستفيد لملف بيانات. ولا يزيد طول اسم الملف عن ثمانية رموز مع اضافة ثلاثة رموز كتوسع لاسم الملف (لاحقة). وتفصل اللاحقة عن اسم الملف بواسطة نقطة فى الصورة FILENAME.EXT.
Function Keys	مفاتيح وظائف وهى المفاتيح من F1 إلى F10 (أو F12) الموجودة على لوحة المفاتيح. وأحياناً تتحدد لها بعض الوظائف بواسطة نظم البرامج.
Hardware	نظم مكونات وهى المعدات التى يتكون منها الكمبيوتر.
Literals	ثوابت وهى قيم لا تستخدم كتمثيل لقيم أخرى، مثال ذلك 1 أو 2 أو 4444 أو "Newark".
Logged drive or directory	وحدة مفتوحة أو دليل مفتوح وهى الوحدة الحالية أو دليل القرص الحالى.
Memory	ذاكرة وهى منطقة تخزين مؤقتة للكمبيوتر. وتستخدم معظم أجهزة الكمبيوتر ذاكرة من أشباه الموصلات ويشار إليها بأنها ذاكرة اتصال عشوائى أو RAM. ويفقد الكمبيوتر المعلومات المخزنة فى الذاكرة عندما يقطع عنه التيار. وتحمل برامج الكمبيوتر وتقيم فى ذاكرة الكمبيوتر عندما يتم تنفيذها.
Module	جزء ويشير فى بيسك السريع إلى مجموعة من أسطر برنامج تخزن فى ملف منفصل.

الاصطلاح	التعريف
Nesting	تداخل وهى طريقة لأحد تكوينات التحكم الذى يحتوى على تكوين تحكم آخر يشبهه أو من نوع آخر.
Operating system	نظام تشغيل وهو برنامج كمبيوتر مصمم للعمل مع الكمبيوتر كسطح بينى بينه وبين نظم برامج التطبيقات.
Pathname	اسم مسار وهو اسم مشغل أو دليل يخزن فيه الملف.
Prompt	ملقن وهو عرض للشاشة يحدد أن النظام معد لإجراء معين من المشغل أو لتقديم معلومات إلى المشغل.
Pull-down Menu	قائمة السحب لأسفل وهى نوع من القوائم يعلق لأسفل من اختيار قائمة أخرى.
RAM	ذاكرة اتصال عشوائى. انظر Memory.
Reseeding	اعادة وضع قيمة للأساس وهى وضع قيمة ابتدائية لمنتج أرقام عشوائية. وهى مثل seeding.
Root Directory	دليل الجذر وهو الدليل الأصلى أو أعلى دليل موجود على القرص.
Run-time Error	خطأ وقت التنفيذ وهو خطأ فى البرنامج يحدث أثناء تنفيذ البرنامج.
Scrolling	دوران وهو حركة أفقية أو رأسية لأحد النصوص على الشاشة.
Software	نظم برامج وهى البرامج (تعليمات الكمبيوتر) وملفات البيانات المقيمة على وسط ممغنط أو على أوراق أو فى نظم مكونات الكمبيوتر.
Stack space	مكان الرصة وهى منطقة فى الذاكرة تخزن فيها تعليمات البرنامج.
Statement Block	مجموعة عبارات وهى مجموعة من أسطر البرنامج محصورة داخل أحد تكوينات التحكم.
Subdirectory	دليل جزئى وهو دليل أو مسار ملف ينتمى إلى دليل مرتفع المستوى.
Symbolic Constant	ثابت رمزى وهو اسم متغير فى بيسك السريع يشير إلى قيمة لاتتغير أثناء تنفيذ البرنامج.
Toggles	وظائف يمكن أن تحول من الوضع on إلى الوضع off والعكس.
Tracing	تتبع وهى عملية يمكنك أن ترى بواسطتها ما يحدث عند كل خطوة أثناء تنفيذ البرنامج.

الاصطـــلاح	التعـــريـف
Watch Points	نقـاط مـلاحظة وهـى مـواقـع مـحـددة فى البـرنامـج يتم مـلاحظتـهـا لتصحيح الأخطاء أو لتحسين الأداء.
Wildcard	رمز خاص يمثل أى قيمة قانونية. يستخدم رمز * فى تمثيل أى اسم ملف قانونى فى DOS.

ملحق B

استخدام منقح بيسك السريع

الجزء الرئيسى لبيئة تطوير بيسك السريع هو المنقح القوى والذى له شاشة ذكاء كاملة. ويقدم هذا الملحق عرضاً عاماً للمنقح. ويعودك هذا الملحق على سمات منقح بيسك السريع التاليـة :

- كتابة برنامج.
- تنفيذ برنامج.
- حفظ برنامج.
- استخدام صناديق الحوار.
- تحميل برنامج.
- كتابة برامج فرعية.
- الانتقال بين البرامج الفرعية.
- القص واللصق.
- الايجاد والاستبدال.
- التتبع وتحديد نقاط الملاحظة.
- مساعدة حساسة للمحتوى.
- تحميل أجزاء أخرى.
- تنقيح نوافذ مجزأة.
- نافذة فورية.
- قشرة DOS
- أوامر أخرى من بيسك السريع.

يسمح لك بيسك السريع بكتابة برامج وتنفيذها وتصحيحها ويسمح لك بملفات رؤية وتتبع تنفيذ البرامج وتنقيح نماذج متعددة وعمل تجزئة لتنقيح الشاشة والتأكد من التكوين وقواعده أثناء ادخال البرنامج. ويقوم بيسك السريع بكتابة الكلمات المحجوزة بحروف كبيرة واتمام التنقيط تلقائياً لتسهيل ادخال البرنامج والتأكد من الأخطاء.

بدء بيسك السريع :

طريقة بدء بيسك السريع هى كما يلى :

١ - إذا كان لديك نظام قرص صلب تأكد أنك فى دليل QB\ (أى أكتب CD\QB واضغط على مفتاح الادخال).

٢ - إذا كان لديك نظام به مشغلا أقراص مرنة، ادخل قرص البرنامج فى المشغل A وادخل قرصاً مشكلاً وفارغاً فى المشغل B، ثم انتقل للعمل على المشغل B.

٣ - إذا كنت تستخدم قرصاً صلباً اكتب QB من عند ملقن DOS واضغط على مفتاح الادخال. وإذا كنت تستخدم نظاماً به مشغلا أقراص مرنة اكتب A:QB واضغط على مفتاح الادخال.

لاحظ شاشة بيسك السريع التى تظهر. تسمح لك خيارات سطر الأمر بتوفير معلومات إضافية تؤثر على كيفية تنفيذ بيسك السريع. وخيارات سطر الأمر التالية يمكن أن تضاف عندما تبدأ بيسك السريع.

الخيــار	الوصـــف
/run source file	تحميل ملف برنامج المصدر وتنفيذه. ملف المصدر source file هو اسم الملف الذى تريد تنفيذه.
/b	يستخدم موجهاً أبيض وأسود مع بطاقة رسومات ملونة. فإذا كانت لديك بطاقة لون واحد فلن تحتاج إلى استخدامه.
/g	للأجهزة التى بها رسومات ملونة ويقلل هذا الخيار من الترددات أثناء دوران النص.
/h	يستخدم أعلى ثبات ممكناً مع نظم المكونات الممكنة. فإذا كانت لديك بطاقة رسومات معززة وموجه رسومات معززة فيغير هذا الخيار العرض إلى 43 سطراً و 80 عموداً.
/c: buffer size	يحدد حجم الذاكرة الاحتياطية للاتصالات. ولها قيمة تقليدية 512 بايت وأقصى حجم هو 32,767 بايت.
/1 library name	يحمل مكتبة Quick محددة.

الخيـار	الوصــــف
/mbf	يتسبب فى أن بيسك السريع يستخدم تشكيل ميكروسوفت الثنائى Microsoft Binary Format بدلاً من تشكيل اعداد IEEE.
/ah	يسمح بسجلات منظومات ديناميكية وسلاسل ثابتة الطول وبيانات عددية تكون أكبر من 64 كيلوبايت.
/cmd string	يستخدم سلسلة string لدالة COMMAND$. ويجب أن يكون آخر عنصر فى السطر.

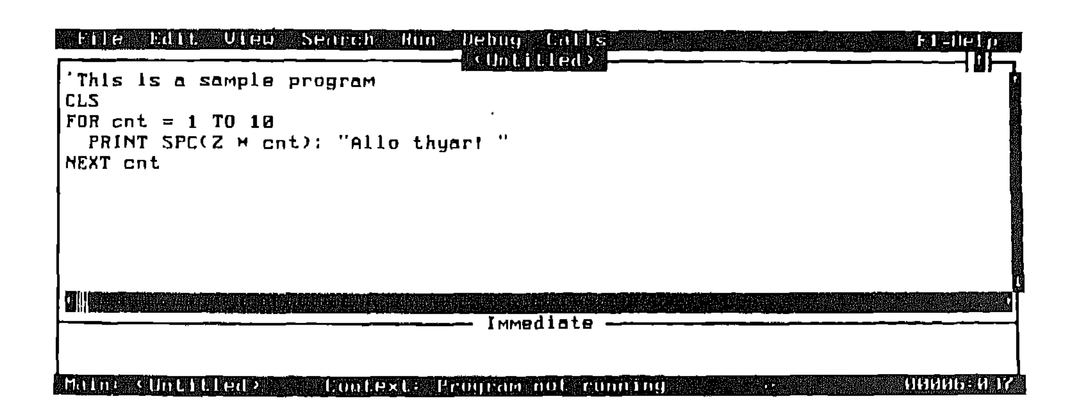
كتابة برنامج

هذا القسم يقدم كيفية كتابة برنامج باستخدام منقح بيسك السريع. نفذ الأنشطة التالية :

١ – ابدأ بيسك السريع كما سبق ذكره فى القسم السابق.

٢ – لاحظ وميض نقطة البداية فى الركن العلوى الأيسر من الشاشة. وهذا يعنى أن بيسك السريع ينتظر منك أن تفعل شيئاً.

٣ – ابدأ بكتابة البرنامج التالى. واضغط على مفتاح الادخال بعد نهاية كل سطر.



لاحظ أن cls و to و print و spc و next تصبح مكتوبة بحروف كبيرة بمجرد انتهائك من كتابة السطر والضغط على مفتاح الادخال. وهذا لأن منقح بيسك السريع ذكى ويميز كلمات بيسك السريع المحجوزة. والخطوات التالية توضح امكانيات أخرى لمنقح بيسك السريع الذكى.

٤ - استخدام مفاتيح نقطة البداية للانتقال إلى علامات التنصيص المزدوجة المستخدمة فى النهاية. (اضغط على Del لحذف علامة تنصيص النهاية).

لاحظ أن بيسك السريع يضيف علامة تنصيص مزدوجة إلى السطر لإتمام التكوين.

٥ - الكلمة "cnt" موجودة فى سطرين، انقل نقطة البداية إلى أقرب واحدة. غير الكلمة cnt إلى Cnt. انقل نقطة البداية إلى سطر أخر.

لاحظ أن بيسك السريع يغير الحدوث الآخر لـ cnt كذلك.

وفيما يلى قائمة بأوامر التنقيح المتاحة فى بيسك السريع :

المفتاح	الاجـــراء
السهم الأيسر أو Ctrl-S	ينقل نقطة البداية خانة واحدة لليسار.
السهم الأيمن أو Ctrl-D	ينقل نقطة البداية خانة واحدة لليمين.
السهم الأيسر -Ctrl أو Ctrl-A	ينقل نقطة البداية كلمة واحدة لليسار.
السهم الأيمن -Ctrl أو Ctrl-F	ينقل نقطة البداية كلمة واحدة لليمين.
السهم العلوى أو Ctrl-E	ينقل نقطة البداية سطراً واحداً لأعلى.
السهم السفلى أو Ctrl-X	ينقل نقطة البداية سطراً واحداً لأسفل.
Home أو Ctrl-O-S	ينقل نقطة البداية لبداية السطر.
End أو Ctrl-Q-D	ينقل نقطة البداية لنهاية السطر.
Ctrl-Enter أو Ctrl-J	ينقل نقطة البداية لبدء سطر جديد.
PgUp أو Ctrl-R	ينقل نقطة البداية صفحة لأعلى.
PgDn أو Ctrl-C	ينقل نقطة البداية صفحة لأسفل.
Ctrl-W	يتحرك حركة دائمة لأعلى مع ثبات نقطة البداية فى موقعها.
Ctrl-Z	يتحرك حركة دائمة لأسفل مع ثبات نقطة البداية فى موقعها.
Ctrl-Q-E	ينقل نقطة البداية إلى قمة النافذة.
Ctrl-Q-X	ينقل نقطة البداية إلى قاعدة النافذة.

المفتاح	الاجـــراء
Ctrl-PgUp	ينقل نقطة البداية لليسار بمقدار نافذة واحدة.
Ctrl-PgDn	ينقل نقطة البداية لليمين بمقدار نافذة واحدة.
Ctrl-Q-R أو Ctrl-Home	ينقل نقطة البـداية إلى بداية جزء جـديد أو بداية إجراء جديد.
Ctrl-Q-C أو Ctrl-End	ينقل نقطة البداية إلى نهاية جزء جديد أو نهاية اجراء جديد.
Ctrl-K-number	وضـع علامات (0-3).
Ctrl-Q-number	الانتقال إلى علامات (0-3).
Ctrl-V أو Ins	تغيير حالة الادخال من on إلى off والعكس.
End-Enter	ادخال سطر من أسفل.
Home-Ctrl-N	ادخال سطر من أعلى.
Shift-Ins	ادخال من لوحة القص.
Tab	عمل الجداول.
Ctrl-P أو Ctrl-any key	لطباعة Ctrl-any key.
سهم أيسر -Shift	لاختيار رمز على اليسار.
سهم أيمن -Shift	لاختيار رمز على اليمين.
سهم سفلى -Shift	لاختيار السطر الحالى.
سهم علوى -Shift	لاختيار السطر الأعلى.
Shift-Ctrl-Left	لاختيار كلمة من على اليسار.
Shift-Ctrl-Right	لاختيار كلمة من على اليمين.
Shift-PgUp	لاختيار بقية الشاشة العلوية.
Shift-PgDn	لاختيار بقية الشاشة السفلية.
Shift-Ctrl-Home	لاختيار بداية جزء أو بداية أجزاء.
Shift-Ctrl-End	لاختيار نهاية جزء أو نهاية أجزاء.
Ctrl-Y	لحذف سطر أو حفظه فى لوحة القص.
Ctrl-Q-Y	للحـذف حـتى نهـاية السطر مع الحـفظ فى لوحـة القص.

المفتـاح	الاجـــراء
Bksp أو Ctrl-H	لحذف الرمز الأيسر.
Del أو Ctrl-G	لحذف الرمز الموجود عند نقطة البداية.
Ctrl-T	لحذف كلمة من ناحية اليمين.
Del	لحذف النص المختار دون حفظه فى لوحة القص.
Shift-Del	لحذف النص المختار مع حفظه فى لوحة القص.
Shift-Tab	لترحيل كل الأسطر المختارة على هيئة جدول.
Ctrl-Ins	لنسخ النص المختار وحفظه فى لوحة القص.

تنفيذ البرنامج

١ – لتنفيذ البرنامج الذى سبق لك اعداده فى القسم السابق اضغط على Shift-F5. لاحظ أنه لا توجد خطوة لترجمة البرنامج قبل تنفيذه. وهذا لأن البرنامج تتم ترجمته أثناء كتابتك له. وفيما يلى شاشة المخرجات.

```
Allo thyar!
  Allo thyar!
    Allo thyar!
      Allo thyar!
        Allo thyar!
          Allo thyar!
            Allo thyar!
              Allo thyar!
                Allo thyar!
                  Allo thyar!

Press any key to continue
```

٢ – اضغط على أى مفتاح للعودة إلى المنقح.

حفظ البرنامج

يمكن أن يحفظ البرنامج الذى قمت بتنفيذه فى القسم السابق وذلك على قرص لاستخدامه فيما بعد. لعمل ذلك استمر على النحو التالى :

١ - اضغط على Alt. لاحظ أن اختيار القائمة الموجودة على أقصى اليسار يضاء.

٢ - اضغط على مفتاح السهم السفلى. لاحظ قائمة السحب لأسفل مع الخيارات المختلفة لمعالجة الملفات.

٣ - اكتب S لاختيار Save، يظهر صندوق يطلب ادخال اسم الملف.

٤ - اكتب Sample كاسم للملف. يدخل بيسك السريع التوسع BAS تلقائياً.

ملاحظة

إذا لم ترغب فى اضافة توسعات على أسماء الملفات اكتب نقطة بعد اسم الملف فوراً (مثل SAMPLE.).

٥ - اضغط على Tab مرتين. لاحظ أن صندوق الحفظ يضاء.

٦ - اضغط على قضيب المسافات. لاحظ أن الملف يحفظ.

لقد سمحت فى الخطوة الخامسة بحفظ الملف كأحد ملفات بيسك السريع وهذا يسمح بتحميله بطريقة أسرع داخل المترجم. وهناك طريقة أخرى لحفظ الملف بحيث يمكن قراءته عن طريق برنامج آخر. ولتجربة هذه الطريقة استمر كما يلى :

١ - اتبع الخطوات من 1 إلى 4 التى سبق ذكرها.

٢ - اضغط على Tab ثم على مفتاح السهم السفلى. لاحظ أن الملف يذكر أنه يحفظ فى صورة ASCII.

٣ - اضغط على Tab ثم اضغط على قضيب المسافات لحفظه.

استخدام صناديق الحوار

تحدث الحركة داخل كل صناديق الحوار dialog boxes (الصناديق التى تظهر عندما تجرى اختياراً من اختيارات القائمة وفى حالات أخرى عندما يتداخل بيسك السريع مع المستفيد) أساساً باستخدام المفاتيح التالية :

المفتـاح	استخدامــه
Tab	يأخـذك من أحـد الاختـيـارات الأسـاسـيـة داخل صندوق الحوار إلى خيار آخر.
Shift-Tab	يعيدك إلى الخيار الذى مررت عليه دون أن تحدده بطريق الخطأ.
مفاتيح الأسهم	يساعدك على الحركة داخل الاختيار الذى حددته وتحديد مفاتيح أو اختيار حالات.
قضيب المسافات	يسـاعدك على وضع المفاتيح فى حالة on أو off واختيار تنفيذ نشاط من صندوق الحوار.
مفتاح الإدخال	يسـتخـدم للاسـتـمـرار فى العملية مع مـفـاتيح وخيارات معينة. فإذا ما غيرت شيئاً واحداً فقط فى صندوق الحـوار فـيـمكنك انهـاء التـداخل والاستمرار مع العملية دون أن تتحرك خلال كل خيارات هذا الصندوق الأخرى.

هذه المفاتيح وهذه المعلومات مقدمة هنا بحيث يمكنك أن تجربها وتعتاد على استخدام صناديق الحوار حيث إنها جزء متكامل من أجزاء بيئة تطوير بيسك السريع.

تحميل البرنامج

هذا القسم يتعامل مع كيفية تحميل ملف موجود فى بيسك السريع.

١ - اضغط على Alt ثم على السهم السفلى.

٢ - اكتب O لفتح ملف. لاحظ صندوق Load مع سرد الدليل الحالى.

٣ - اضغط على Tab للانتقال إلى الدليل.

٤ - استخدم مفاتيح الأسهم للانتقال حول الدليل.

نصيحة : هناك طريقة أخرى لعمل ذلك وهى كتابة أول حرف من اسم الملف. ويأخذك هذا إلى أول اسم ملف يبدأ بهذا الحرف. مثال ذلك إذا كانت لديك ثلاثة ملفات تبدأ بنفس الحرف مثل

Factor. bas و Fun.bas و Func.bas فكتابة F أكثر من مرة واحدة تأخذك فى أول مرة إلى Factor.bas وفى المرة الثانية إلى Fun.bas وفى المرة الثالثة إلى Func.bas ثم تعيدك بعد ذلك إلى Factor.bas مرة أخرى. وهذه طريقة أسهل لاختيار الملفات إذا لم تكن متأكداً من الهجاء الصحيح لأسماء الملفات.

٥ – عندما تكون نقطة البداية على الملف SAMPLE. BAS اضغط على مفتاح الادخال.

كتابة برامج فرعية

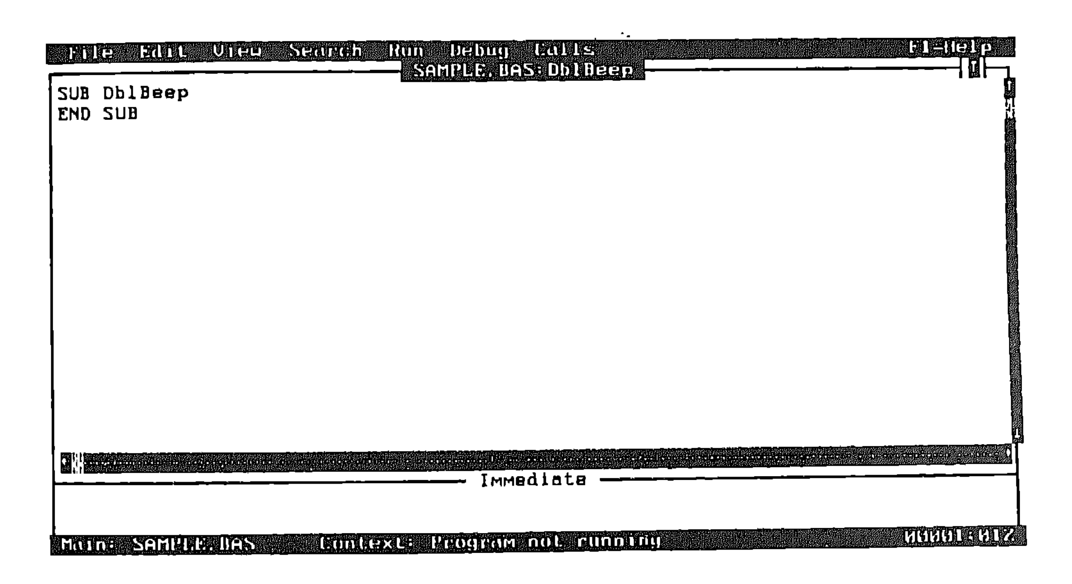
هذا القسم يوضح كيفية عزل بيسك السريع للبرامج الفرعية من الجزء المنادى وحفظها منظمة بحيث يمكنك نقل أحدها إلى الآخر أثناء التنقيح. ولمحاولة عمل ذلك استمر على النحو التالى :

١ – حمل البرنامج SAMPLE.BAS مستخدماً الاجراءات التى سبق ذكرها فى القسم السابق.

٢ – اضغط على Alt-E للذهاب إلى قائمة Edit.

٣ – أكتب S لاختيار SUB جديد. لاحظ النافذة التى تظهر سائلة عن اسم البرنامج الفرعى.

٤ – أكتب اسم البرنامج الفرعى Db1Beep واضغط على مفتاح الادخال.



٥ – لاحظ الشاشة الفارغة تحت عنوان SAMPLE.BAS:DblBeep ومجموعة SUB و ENDSUB.

٦ – انقل نقطة البداية إلى نهاية أول سطر واضغط على مفتاح الادخال. أكتب Beep: Beep.

الانتقال بين البرامج الفرعية

لتعريف البرنامج الفرعى DblBeep داخل البرنامج SAMPLE.BAS استمر على النحو التالى :

١ – اضغط على Shift-F2 للذهاب إلى الجزء الرئيسى SAMPLE.BAS. يسمح لك Shift-F2 بالانتقال من برنامج فرعى لآخر.

٢ – اكتب DblBeep قبل وبعد دورة FOR NEXT.

٣ – اضغط على Alt-F واكتب S لحفظ البرنامج. لاحظ عبارة DECLARE التى يدخلها بيسك السريع للبرنامج الفرعى الجديد DblBeep. وتحدد الأقواس الفارغة أنه لا توجد هناك مؤشرات للبرنامج الفرعى DblBeep.

٤ – اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ الصوت المزدوج للصفارة قبل وبعد الطباعة.

لرؤية كيفية تناسق SUB مع البرنامج افعل ما يلى :

١ – انتقل إلى قائمة View . ويمكنك أن تفعل ذلك بأى من الطريقتين التاليتين :

– اضغط على Alt-F ومفتاح السهم الأيمن مرتين للانتقال إلى قائمة View

– أو اضغط على Alt-V.

٢ – اكتب S لـ Subs.

وهذا يعطى نافذة تعرض كل الأجزاء وكل البرامج الفرعية المحملة حالياً فى الذاكرة وكيفية ارتباطها ببعضها البعض. من هذه النافذة يمكنك أن تختار برنامجاً فرعياً أو جزءاً لتنقيحه أو لحذفه. اضغط على مفاتيح السهم العلوى والسفلى للانتقال من أحد المحتويات لآخر مع ملاحظة قاعدة الشاشة. يعرض اسم المحتوى مع تحديد ما إذا كان برنامجاً فرعياً أو جزءاً رئيسياً فإذا كان برنامجاً فرعياً فيظهر الجزء الذى ينتمى إليه هذا البرنامج الفرعى.

نصيحة : الضغط على F2 هو طريقة مختزلة للوصول إلى صندوق حوار SUB بدون استخدام قائمة View.

القص واللصق

هذا القسم يصف القص واللصق للأسطر داخل أحد الأجزاء. ابدأ عند صندوق حوار SUB الذى سبق شرحه فى القسم السابق واستمر على النحو التالى :

١ – انقل الاضاءة إلى DblBeep واضغط على مفتاح الادخال ثم انقل نقطة البداية إلى العبارة Beep: Beep.

٢ – اضغط على السهم السفلى -Shirt ولاحظ السطر الذى يضاء.

٣ – اضغط على Ctrl-Ins لحفظه فى لوحة القص. ولوحة القص هى مكان يخزن فيه النص لإجراء عملية لصق.

٤ – انقل نقطة البداية إلى نهاية السطر واضغط على مفتاح الادخال لإنتاج سطر جديد.

٥ – اضغط على Shift-Ins للصق السطر من لوحة القص. لاحظ أن السطر يظهر.

٦ – احفظ البرنامج كما سبق ذكره فى قسم سابق.

يمكن أن تنفذ عملية القص واللصق كذلك على أجزاء من أسطر باستخدام سهم أيمن -Shift وسهم أيسر -Shift.

الايجاد والاحلال

نتعلم من هذا القسم كيفية إيجاد سلسلة رموز فى البرنامج واحلال سلسلة أخرى محلها. كذلك تكون لديك الفرصة فى ممارسة التداخل مع صناديق الحوار. استمر كما يلى :

١ – اضغط على Shift-F2 للذهاب إلى الجزء الرئيسى. انقل بعد ذلك نقطة البداية إلى أول سطر فى البرنامج.

٢ – اضغط على Ctrl-Q-A. يظهر صندوق لكتابة سلسلة البحث.

٣ – اكتب "How Now Brown Cow" واضغط على Tab.

٤ – تكون الآن فى الحقل الذى تريد أن تحدد فيه ما تريد أن تغيره How Now Brown Cow. اكتب "!Don't Frown Beneath The Bough" واضغط على Tab.

٥ – انتقل إلى خيار Search واختر All Modules بالضغط على السهم السفلى ثم الضغط على Tab.

٦ - ضع البحث عند Match Upper/ Lowercase بالضغط على قضيب المسافات.

٧ - اضغط على Tab مرتين لترك حقل Whole Word.

٨ - انتقل إلى Find and Verify واستمر في العملية بالضغط على مفتاح الادخال.

٩ - عندما توجد السلسلة التي يجرى البحث عنها اسمح لها بالتغيير إلى السلسلة الجديدة وذلك بالضغط على مفتاح الادخال كاستجابة لصناديق الحوار.

١٠ - احفظ هذا البرنامج كما سبق ذكره في الأقسام السابقة.

إذا كانت لديك أي مشاكل مع أي خطوة من الخطوات السابقة فيجب أن ترجع إلى قسم استخدام صناديق الحوار.

التتبع وتحديد نقاط ملاحظة

هذا القسم يقدم تمريناً فى تتبع أحد البرامج. يسمح التتبع tracing بتنفيذ برنامج ورؤية مايحدث عندما ينفذ كل أمر من أوامره. وهذا مفيد عندما لا يعمل البرنامج ولا يكون لديك أى مفتاح لمعرفة ما يحدث فيه. ونقاط الملاحظة watch points هى شروط تقوم بوضعها بنفسك لمتغيرات توقف تنفيذ البرنامج عندما تتحقق هذه الشروط. وهذه وسيلة تصحيح أخرى فى بيسك السريع. جرب المثال التالى :

١ – حمل البرنامج SAMPLE.BAS إذا لم يكن محملاً بالفعل.

٢ – اضغط على Alt-D للذهاب إلى قائمة Debug واختر Watchpoint.

٣ – أكتب Cnt = 5 واضغط على مفتاح الادخال. لاحظ أن نافذة الملاحظة تفتح عند قمة الشاشة مع اسم البرنامج والشرط المحدد.

```
SAMPLE.BAS: Cnt = 5
```

٤ – اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ أن البرنامج يتوقف وأن الشرط فى نافذة Watch هو <TRUE>. لاحظ كذلك سطر البرنامج المضاء عندما يتوقف التنفيذ.

٥ – اضغط على F4 لرؤية المخرجات. لاحظ أن أربعة أسطر فقط قد تم طباعتها. اضغط على F4 للعودة إلى نافذة الرؤية View.

٦ – احذف Watchpoint عن طريق اختيار Delete Watch من قائمة Debug.

يمكنك أن تدخل اسم متغير فى نافذة Watch بدون شرط لرؤية القيم التى يفترضها المتغير أثناء تنفيذ البرنامج. استخدم خيار Add Watch من قائمة Debug واستخدم طريقة التنفيذ خطوة خطوة (F8). فيما يلى قائمة بالأوامر التى يمكنك استخدامها فى تتبع تنفيذ البرنامج.

المفتاح	الوصف
F4	الذهاب إلى شاشة المخرجات أو العودة إلى قائمة البرنامج.
F5	تنفيذ من عند العبارة الحالية.
Shift-F5	تنفيذ من البداية.
F7	تنفيذ البرنامج إلى موقع نقطة البداية.

المفتاح	الوصـــف
F8	تنفيذ عبارة البرنامج الحالية مع التتبع. أى تنفيذ كل عبارات البرنامج بما فى ذلك تنفيذ البرامج الفرعية والتتبع.
Shift-F8	التتبع التاريخى للخلف.
F9	تغيير نقاط التقاطع من on إلى off والعكس.
F10	تنفيذ العبارة التالية وتتبع الاجراء. أى تنفيذ كل عبارات البرنامج فى الجزء الرئيسى فقط وليس فى البرامج الفرعية أو تنفيذ التتبع.
Shift-F10	التتبع التاريخى للأمام.

مساعدة حساسة للمحتوى

المساعدة الحساسة للمحتوى context sensitive help هى مساعدة محددة لما تفعله عندما تطلب مساعدة، وفى بيسك السريع يمكنك الاتصال بها عن طريق الضغط على Shift-F1. وهى بسيطة جداً. استمر على النحو التالى :

١ - إذا لم تكن فى الجزء الرئيسى اضغط على F2 واختر SAMPLE.BAS من صندوق الحوار.

٢ - ضع نقطة البداية على عبارة FOR واضغط على Shift-F1.

لاحظ أن النصف العلوى للشاشة قد ظهر فيه توضيح لعبارة FOR.. NEXT. وهى قاعدة الشاشة ترى البرنامج الذى تم تنقيحه بواسطة عبارة FOR.

٣ - يمكنك الحركة حول البرنامج وطلب المساعدة على ذلك بنفس الطريقة. حاول عمل ذلك.

٤ - ضع نقطة البداية على استدعاء SUB لـ DblBeep واطلب المساعدة. لاحظ أن بيسك السريع يفتح نافذة بكل كلمات بيسك السريع المحجوزة ويسألك أن تختار واحدة منها. وهذا لأن DblBeep ليست كلمة بيسك سريع محجوزة. وعلى هذا لا توجد مساعدة متاحة لها وعند ذلك يفعل بيسك السريع الشىء الجيد التالى ويسمح لك باختيار أحد الأشياء. لاختيار واحدة ضع الاضاءة عليها مستخدماً مفاتيح الأسهم واضغط على مفتاح الادخال.

٥ - اضغط على Esc مرتين للعودة إلى قائمة البرنامج.

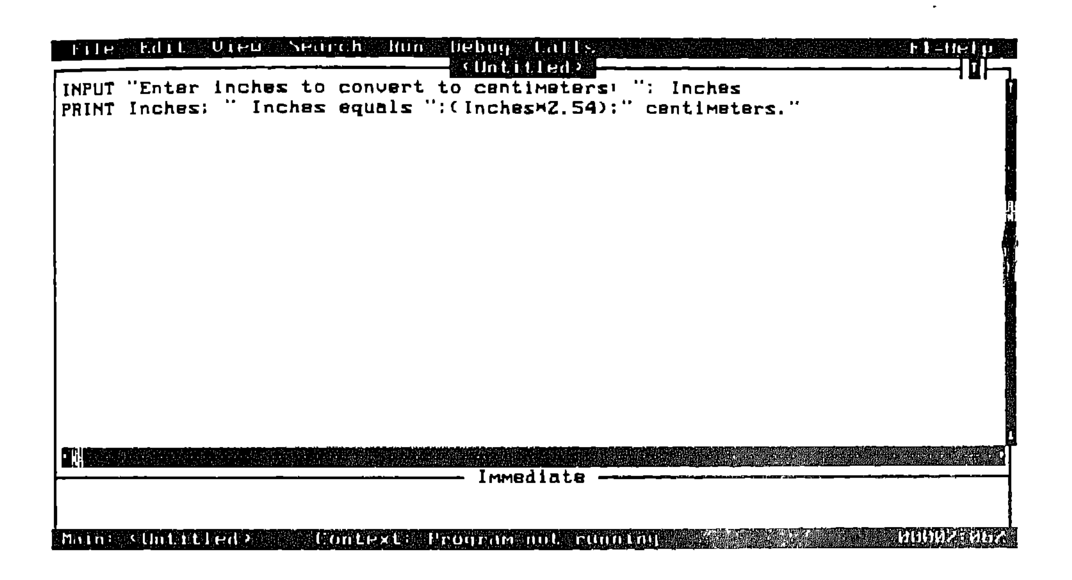
تحميل أجزاء أخرى

فى بيسك السريع الجزء module عبارة عن مجموعة من أسطر البرنامج محفوظة فى ملف منفصل يمكن تنفيذه بنفسه أو مع أجزاء أخرى.

وفى هذا القسم تقوم بكتابة برنامج صغير آخر ثم تحمل البرنامج SAMPLE.BAS فى بيسك السريع دون أن تفقد البرنامج الجديد. استمر على النحو التالى :

١ – أحفظ البرنامج SAMPLE.BAS واختر New من قائمة File بالضغط على Alt-F وكتابةN.

٢ – اكتب البرنامج التالى :



٣ – احفظ هذا البرنامج على أنه ملف ASCII تحت اسم INCH2CM.BAS.

٤ – انتقل إلى قائمة File واختر Load. لاحظ صندوق حوار تحميل الملف مع خيارات الحالة التى تريدها وقائمة الدليل. اجعل الحالة على انها جزء module.

٥ – انتقل إلى الدليل بالضغط على Tab مرتين واختر SAMPLE.BAS واضغط على مفتاح الادخال.

٦ - انتقل إلى قائمة View واكتب S لرؤية كل الأجزاء وكل البرامج الفرعية.

نصيحة : F2 هي طريقة مختزلة لأداء نفس الشيء.

٧ - باستخدام مفاتيح الأسهم اختر SAMPLE.BAS ولاحظ الوصف الموجود في القاعدة.

هذا يبين أنك تستطيع تحميل أكثر من ملف واحد في الذاكرة في نفس الوقت. وحيث إن الجزئين المحملين ليسا مرتبطين مع بعضهما البعض فينفذ أول نموذج تم تحميله فقط عندما تضغط على Shift-F5. ولتنفيذ أي نموذج آخر تم تحميله حدد أن الجزء الرئيسي هو هذا الجزء وذلك من قائمة Run. ولمحاولة ذلك افعل ما يلي :

١ - اضغط على esc للعودة إلى قائمة البرنامج.

٢ - انتقل إلى قائمة Run واختر Set Main Module. لاحظ ظهور صندوق الحوار مع قائمة بكل الأجزاء المحملة.

٣ - باستخدام مفاتيح الأسهم اختر INCH2CM.BAS ثم اضغط على Tab ثم على قضيب المسافات.

٤ - اضغط على Shift-F5 لتنفيذ الجزء الرئيسي الجديد مع ملاحظة أن SAMPLE.BAS لم ينفذ.

٥ - نفذ الخطوات من 1 إلى 3 وحدد أن SAMPLE.BAS هو الجزء الرئيسي.

تنقيح الشاشة المجزأة

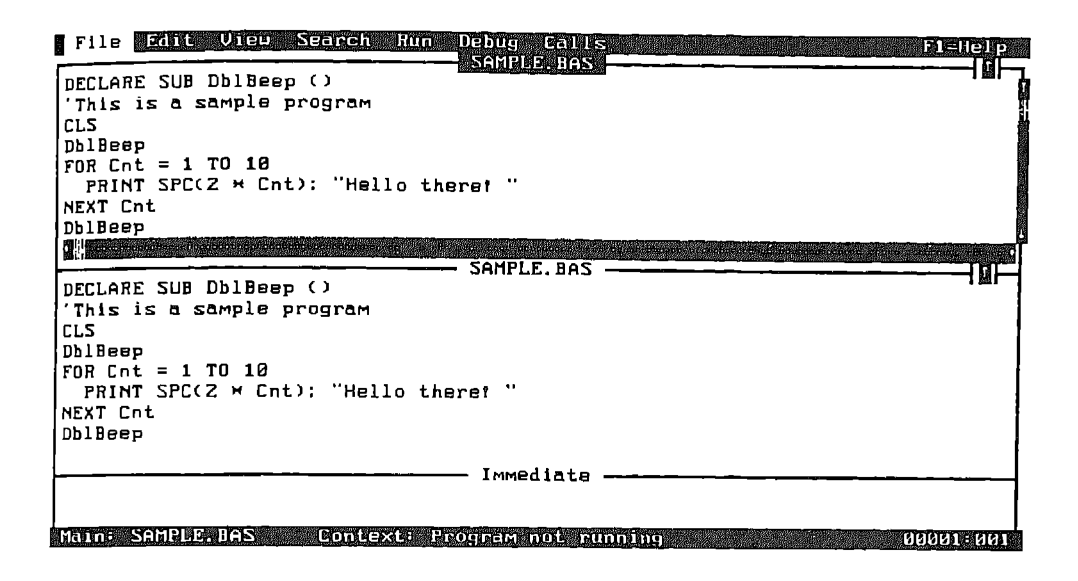
ترى في هذا القسم برنامجين أو جزءين من نفس البرنامج وذلك في نفس الوقت. استمر على النحو التالي :

١ - انتقل إلى قائمة File واختر New. يلغي هذا الخيار كل البرامج الموجودة في الذاكرة بحيث يمكنك أن تكتب برنامجاً جديداً. لاحظ أن بيسك السريع يلقنك بأن الأجزاء المحمّلة حالياً في الذاكرة لم تحفظ. اختر No في صندوق الحوار ثم اضغط على قضيب المسافات.

٢ - انتقل إلى قائمة File واختر Open. حمل البرنامج SAMPLE.BAS.

٣ - اختر قائمة View وأكتب P لترى ثلاث نوافذ على الشاشة. لاحظ الثلاث نوافذ، الأولى منها الجزء الرئيسي والثانية معها نفس الجزء والنافذة الفورية.

٤ – اضـغط على F6 للانتقـال إلى النافذة التـالية. والآن اضـغط على Shift-F2 لرؤية البرنامج الفرعى DblBeep. لاحظ أنك تنظر إلى جزءين مختلفين من نفس البرنامج فى نفس الوقت.



```
File  Edit  View  Search  Run  Debug  Calls                    F1=Help
                              SAMPLE.BAS
DECLARE SUB DblBeep ()
'This is a sample program
CLS
DblBeep
FOR Cnt = 1 TO 10
  PRINT SPC(2 * Cnt); "Hello there! "
NEXT Cnt
DblBeep
                              SAMPLE.BAS
DECLARE SUB DblBeep ()
'This is a sample program
CLS
DblBeep
FOR Cnt = 1 TO 10
  PRINT SPC(2 * Cnt); "Hello there! "
NEXT Cnt
DblBeep
                              Immediate

Main: SAMPLE.BAS      Context: Program not running          00001:001
```

٥ – اضـغط على Shift-F5 للعودة إلى الخلف نافذة واحدة. الضـغط على F6 يأخذك من نافذة إلى أخرى وذلك من النافذة العلوية إلى النافذة السفلية. ويأخذك الضغط على Shift-F6 من النافذة السفلية إلى النافذة العلوية واحدة بعد الأخرى.

٦ – اضـغط على Ctrl-F10 لإعادة النافذة الحالية إلى شاشة كاملة، يمكن عمل ذلك مع أى من الثلاث نوافذ.

٧ – اختر New من قائمة File. استجب كما فعلت فى الخطوة رقم 1 لملقن بيسك السريع.

النافذة الفورية

النافذة الفورية immediate window هى سمة تسمح لك بادخال عبارات برنامج بيسك وتنفيذها بالضغط على مفتاح الادخال. حاول عمل ذلك باتباع الخطوات التالية :

١ – اضغط على F6 للانتقال إلى النافذة الفورية Immediate Window.

٢ - اكتب العبارة التالية :

```
Cls: For i = 1 to 10: Print "Ze emmediate weendow"
```

٣ - اضغط على مفتاح الادخال لتنفيذ عبارة البرنامج، لاحظ أن العبارة تنفذ على الفور.

وفيما يلى خواص النافذة الفورية :

- يمكنك أن تنفذ سطراً فردياً بوضع نقطة البداية على السطر والضغط على مفتاح الادخال، ولاينفذ إلا هذا السطر.
- يمكنك ادخال عدد من السطور لا يزيد عن 10 أسطر ولاتزيد الرموز فى السطر الواحد عن 256 رمزاً. يمكنك كتابة أكثر من عبارة واحدة على نفس السطر. أنظر المثال السابق.
- يمكنك أن تغير الحجم بالضغط على +-Alt للزيادة أو - -Alt للنقصان أو Ctrl-F10 للعودة إلى الشاشة الكاملة.
- يمكن حفظ الشفرة على قرص إلا إذا استخدمت أوامر القص والنسخ Cut and Copy من قائمة Edit.
- يمكن استخدام معظم الدوال والعبارات فى النافذة الفورية. وفيما يلى قائمة بكلمات بيسك السريع المحجوزة غير المسموح بها فى النافذة الفورية.

COMMON	DEF FN	SHARED	$DYNAMIC
DATA	DIM	STATIC	REDIM
$STATIC	OPTION	TYPE	END IF
SUB	END SUB	END DEF	END TYPE
ELSEIF	FUNCTION	DECLARE	$INCLUDE
END FUNCTION	CONST	DEFtype	

وفيما يلى بعض استخدامات النافذة الفورية :

- اختبار عبارات PRINT.

- اختبار الحسابات.

- استدعاءات إجراءات لاختبارها منفصلة.

- تغيير قيم المتغيرات فى تنفيذ البرنامج. ولعمل ذلك نفذ البرنامج من نافذة View وأفصله باستخدام Ctrl-Break وانتقل إلى النافذة الفورية وغير قيمة المتغير وأضغط على F5 للاستمرار فى التنفيذ. والسطر المضاء فى نافذة View هو السطر الذى يتأثر.

- محاكاة أخطاء وقت التنفيذ بتحديد قيمة خطأ لعبارة ERROR.

٤ - اضغط على F6 للعودة إلى نافذة View.

قشرة DOS

هذا القسم يصف اختيار DOS Shell من قائمة File. ويسمح DOS Shell بترك بيسك السريع لحظيا والذهاب لتنفيذ أوامر DOS ثم العودة إلى بيسك السريع دون فقدان بيانات ولمحاولة ذلك استمر على النحو التالى :

١ - انتقل إلى قائمة File واختر DOS Shell. تظهر التعليمات الخاصة بالعودة إلى بيسك السريع وملقن DOS.

```
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow
Ze emmediate weendow

Type EXIT to return to QuickBASIC


The IBM Personal Computer DOS
Version 3.30 (C)Copyright International Business Machines Corp 1981, 1987
             (C)Copyright Microsoft Corp 1981, 1986

C:\QB>
```

٢ - اكـتب DIR/W عند ملقن DOS واضـغط على مـفـتـاح الادخـال. لاحظ أن الدليل تظهـر محتوياته.

٣ - اكتب EXIT واضغط على مفتاح الادخال. لاحظ ظهور العبارة المعتادة "Press any key to continue" من بيسك السريع. افعل ذلك للعودة إلى بيسك السريع.

٤ - اضغط على Alt-F وأكتب X للخروج من بيسك السريع.

أوامر بيسك سريع أخرى

فيما يلى أوامر أخرى متاحة فى بيسك السريع :

المفتاح	الغـرض منـه
F1	يعطى معلومات مساعدة عامة.
F2	يسـرد الملفات المحملة (SUB و FUNCTION و module و include و document).
F3	يعيد آخر أمر ايجاد.
F4	يبين شاشة المخرجات.
F5	يستمر مع تنفيذ البرنامج.
F6	ينقل إلى النافذة التالية.
F7	ينفذ البرنامج إلى موقع نقطة البداية.
F8	ينفذ العبارة التالية مع التتبع خلال الاجراءات.
F9	ينقل من وإلى نقاط التقاطع.
F10	ينفذ العبارة التالية مع التتبع خلال الإجراء.
Shift-F6	يعيد إلى النافذة السابقة.
Ctrl-F2	يعيد إلى البرنامج السابق.
Ctrl-F5	يعيد الشاشة الكاملة إلى حجمها السابق.
Ctrl-F10	يجعل النافذة النشطة كاملة الحجم.
Ctrl-\	يحضر النص المختار.

المفتاح	الغــرض منــه
Ctrl-Q-A	يعيد الايجاد والاستبدال.
Alt-Bksp	يلغى آخر تنقيح.
Esc	يغلق آخر نافذة.

انتقل إلى الدرس الثالث للاستمرار فى تسلسل التعلم.

ملحق (C)

جدول ASCII وكلمات بيسك السريع المحجوزة

جدول ASCII

Regular ASCII Chart (character codes 0 - 127)

Code	Char	Name	Code	Char	Name	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
000		(nul)	016	►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺	(soh)	017	◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☻	(stx)	018	↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥	(etx)	019	‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦	(eot)	020	¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	♣	(enq)	021	§	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠	(ack)	022	▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	•	(bel)	023	↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	◘	(bs)	024	↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009		(tab)	025	↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010		(lf)	026		(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂	(vt)	027	←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	♀	(np)	028	∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	¦
013		(cr)	029	↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	♫	(so)	030	▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	☼	(si)	031	▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

Extended ASCII Chart (character codes 128 - 255)

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	β	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

كلمات بيسك السريع المحجوزة

الكلمات المحجوزة هى كلمات تحجز كأوامر للغة البرمجة. ولا يمكن استخدام هذه الكلمات فى أى شىء آخر غير الفرض الذى سبق تحديده لها. وتسمى الكلمات المحجوزة reserved words بأنها كلمات رئيسية key words أو بأنها أفعال اللغة language verbs كذلك. وفيما يلى قائمة بكلمات بيسك السريع المحجوزة.

ABS
ACCESS
ALIAS
AND
ANY
APPEND
AS
ASC
ATN
BASE
BEEP
BINARY
BLOAD
BSAVE
BYVAL
CALL
CALLS
CASE
CDBL
CDECL
CHAIN
CHDIR
CHR$
CINT
CIRCLE
CLEAR
CLNG
CLOSE
CLS
COLOR
COM
COMMAND$
COMMON
CONST
COS
CSNG
CSRLIN
CVD
CVDMBF
CVI
CVL
CVS
CVSMBF
DATA
DATE$
DECLARE
DEF
DEFDBL
DEFINT
DEFLNG
DEFSNG
DEFSTR
DIM
DO
DOUBLE
DRAW
ELSE
ELSEIF
END
ENDIF
ENVIRON
ENVIRON$
EOF
EQV
ERASE
ERDEV
ERDEV$
ERL
ERR
ERROR
EXIT
EXP
FIELD
FILEATTR
FILES
FIX
FOR
FRE
FREEFILE
FUNCTION
GET
GOSUB
GOTO
HEX$
IF
IMP
INKEY$
INP
INPUT
INPUT$
INSTR
INT
INTEGER
IOCTL
IOCTL$
IS
KEY
KILL
LBOUND
LCASE$
LEFT$
LEN
LET
LINE
LIST
LOC
LOCAL
LOCATE
LOCK
LOF
LOG
LONG
LOOP
LPOS
LPRINT
LSET
LTRIM$
MID$
MKD$
MKDIR
MKDMBF$
MKI$
MKL$
MKS$
MKSMBF$
MOD
NAME
NEXT
NOT
OCT$
OFF
ON
OPEN
OPTION
OR
OUT
OUTPUT
PAINT
PALETTE
PCOPY
PEEK
PEN
PLAY
PMAP
POINT
POKE
POS
PRESET
PRINT
PSET
PUT
RANDOM
RANDOMIZE
READ
REDIM
REM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RMDIR
RND
RSET
RTRIM$
RUN
SADD
SCREEN
SEEK
SEG
SELECT
SETMEM
SGN
SHARED
SHELL
SIGNAL
SIN
SINGLE
SLEEP
SOUND
SPACE$
SPC
SQR
STATIC
STEP
STICK
STOP
STR$
STRIG
STRING
STRING$
SUB
SWAP
SYSTEM
TAB
TAN
THEN
TIME$
TIMER
TO
TROFF
TRON
TYPE
UBOUND
UCASE$
UNLOCK
UNTIL
USING
VAL
VARPTR
VARPTR$
VARSEG
VIEW
WAIT
WEND
WHILE
WIDTH
WINDOW
WRITE
XOR

ملحق (D)

مترجم وواصل سطر الأوامر

يمكن أن تترجم البرامج التى كتبتها باستخدام بيسك السريع بحيث يمكنك تنفيذها من ملقن DOS بدون استخدام بيسك السريع. احدى طرق عمل ذلك هى انتاج ملف EXE. من بيئة بيسك السريع. الطريقة الأخرى هى استخدام مترجم وواصل سطر أوامر البيسك. ويصف هذا الملحق طريقة ترجمة سطر الأوامر. وفيما يلى قائمة بالملفات المستخدمة فى هذه الطريقة مع وصف موجز لها.

الملف	الوصف
BC.EXE	هذا هو مترجم سطر الأوامر ويستخدم عند ترجمة البرامج المكتوبة بمنقح آخر أو ترجمة برامج أكبر من أن تتم ترجمتها داخل الذاكرة.
LINK.EXE	هذا هو واصل Microsoft Overlay Linker المستخدم فى توصيل ملفات Obj. مع مكتبات وقت التنفيذ اللازمة.

الترجمة بواسطة BC. EXE

هذا القسم يصف كيفية استخدام برنامج BC.EXE فى ترجمة ملفات المصدر. وتكوين مترجم سطر الأوامر BC هو كما يلى :

```
BC srcfile,objfile,listfile optionlist;
```

جزء scrfile هو ملف مصدر بلغة البيسك. وجزء objfile هو اسم الملف الذى تذهب إليه الشفرة التنفيذية وعندما يحذف هذا الجزء فيكون اسم ملف التنفيذ هو نفس اسم ملف المصدر ولكن باتساع Obj.. وجزء listfile هو اسم الملف الذى تذهب إليه قائمة برنامج المصدر وعندما يحذف هذا الجزء فلا يتم انتاج ملف بقائمة المصدر. وجزء optionlist هو قائمة بخيارات ترجمة

تكون مهمة عند استخدام بعض سمات بيسك السريع. وفيما يلى قائمة خيارات سطر الأمر BC.EXE المختلفة :

الخيار	الوصف
/a	لانتاج شفرة أشياء غير مجمعة وتوضيح شفرة التجميع التى تنتج لكل سطر من أسطر البرنامج.
/ah	يسمح بمنظومات ديناميكية لسجلات وبسلاسل ثابتة الطول وأنواع عددية يكون حجمها أكبر من 64 كيلوبايت.
/c:bufsize	تحدد حجم الذاكرة الاحتياطية للاتصالات وحدها الأقصى هو 32,767 وقيمتها التقليدية هى 256 بايت. وهى ذاكرة احتياطية للاستقبال فقط.
/d	تنتج شفرة تصحيح للتأكد من أخطاء وقت التنفيذ والتمكين من عمل Ctrl-Break.
/e	تستخدم فى البرامج مع عبارات ON ERROR و RESUME وتحدد وجود هذه العبارات.
/mbf	تعالج الاعداد فى تشكيل IEEE على أنها أعداد لها شكل ميكروسوفت الثنائى. تعالج الدوال CVD و CVS و MKD# و MKS$ كدوال CVDMBF و CVSMBF و MKDMBF$ و MKSMBF$.
/o	تعرض بـ BCOM40. LIB بدلاً من BRUN40. LIB. وتستخدم عندما يترجم البرنامج بحيث لا يكون وقت التنفيذ الذى يدعم مكتبة BRUN40. BAS مطلوباً. ويكون البرنامج المترجم أطول.
/r	تخزن صف منظومة رئيسياً وتستخدم مع مقاطع أخرى مكتوبة بلغات أخرى إذا كانت تخزن صف منظومة رئيسياً.
/s	تكتب سلسلة ثابتة فى ملف التنفيذ بدلاً من جدول الرموز. تستخدم مع البرامج التى تستخدم كميات من ثوابت السلاسل.

الخيـار	الوصـف
/v	تمكن من اصطياد الأحداث وتستخدم مع عبارة ON event.
/w	نفس الشيء مثل v/ مع التأكد من الأحداث بين العبارات.
/x	تحدد أن البرنامج يحتوى على عبارات ON ERROR و RE-SUME و RESUME NEXT و RESUMED.
/zd	تنتج ملف تنفيذ بسجلات لعدد الأسطر تناظر ملف المصدر. تستخدم عندما ترغب فى استخدام معيد التصحيح الرمزى لميكروسوفت Microsoft Symbolic Debugger (SYMDEB).
/zi	تنتج شفرة تنفيذ متوافقة مع معيد تصحيح شفرة رؤية ميكروسوفت Microsoft Code View.

التوصيل : هذا القسم يصف استخدام واصل ميكروسوفت Microsoft Overlay Linker (LINK.EXE). ويمكن استدعاء برنامج الوصل بإحدى الطرق التالية :

- كتابة عبارة LINK باستخدام التكوين التالى :

```
LINK objfile.exefile.mapfile.lib linkopts;
```

حيث objfile هو الملف الذى ينتج بواسطة BC.EXE وجزء exefile هو المكان الذى يذهب إليه البرنامج الذى يتم توصيله (وعندما يحذف هذا الجزء يكون للملف نفس الاسم مثل ملف obj. ولكن يكون اتساعه هو exe.) وجزء mapfile يرسم قائمة إذا ما اخترته أما جزء lib. فهو مكتبة واحدة أو أكثر يراد ايصالها بملف التنفيذ لانتاج ملف exe. أما جزء inkopts فهو قائمة خيارات الاتصال.

- كتابة LINK والضغط على مفتاح الادخال والاجابة عن الأسئلة.

- اعداد ملف بالاستجابات المناسبة بنفس الترتيب مثل الأسئلة وكتابة Link & responsefile عند ملقن DOS حيث responsfile هو اسم الملف للاستجابات التى انتجته.

وفيما يلى قائمة بخيارات التوصيل المختلفة التى يقبلها LINK. EXE.

الخيـار	الوصـف
/HE	رؤية قائمة الخيارات المتاحة.
/PAU	تستخدم عندما تكون المقايضة للفرص مشمولة وتتسبب فى إيقاف التوصيل قبل كتابة ملف التنفيذ على القرص.
/I	تعرض معلومات عن عملية التوصيل.
/B	تتسبب فى أن الواصل لا يعطى ملقناً إذا لم يستطع أن يجد مكتبة أو ملف تنفيذ. ويحدث التوصيل فى صورة تشغيل دفعة.
/Q	تدمج ملفات التنفيذ المحددة لمكتبة سريعة. يجب أن يدعم BQLB40.LIB كأحد مكتباته.
/E	تحذف تسلسل بايت مكررة ويجعل الشفرة أكثر ايجازاً لجعل التحميل أكثر سرعة.
/NOP	تغلق عمل تعبئة قطاع الشفرة.
/NOD	لا يتم البحث فى المكتبات التى تأتى بعده لحل resolve الاختلافات الخارجية. يستخدم لتجنب تحميل مكتبات نمطية مكررة.
/SE	للتحكم فى عدد القطاعات التى يسمح الواصل بها للبرنامج. القيمة التقليدية هى 128 ويمكن أن تصل إلى 1024 كحد أقصى.
/M	لإنتاج ملف خريطة.
/LI	تشمل أرقام السطور والعناوين المصاحبة لبرنامج المصدر فى ملف الخريطة.
/PAC: number	تتسبب فى أن الواصل يجمع قطاعات الشفرة المتجاورة بحيث أنها تقتسم نفس عنوان القطاع. والرقم المحدد هو حجم المجموعة وله قيمة تقليدية 65,530.
/CO	تنتج ملف Exe. يمكن تصحيحه باستخدام Code View De-bugger ويجب أن يستخدم خيار zi/ أثناء الترجمة.

الخيـار	الوصـف
/NOI	تتسبب فى أن الواصل يميز بين الحروف الكبيرة والحروف الصغيرة. لا تستخدمه.
/CP	تضع أقصى عدد للمقاطع (16 بايت) يتطلبه البرنامج عند تحميله. ويراقب الذاكرة أثناء تنفيذ البرنامج.
/DO	تجبر ترتيب القطاعات طبقاً لمنتجات لغة Microsoft مرتفعة المستوى التقليدية.
/ST: number	تضع حجم للرصة ويقع بين 1 و 65,535 وله قيمة تقليدية 2 كيلوبايت.
/DS	تحمل كل البيانات بدءاً عند النهاية المرتفعة لقطاع البيانات التقليدى.
/HI	تضع البرامج مرتفعة بقدر الإمكان فى الذاكرة.
/NOG	تهمل مصاحبات المجموعات عند تحديد عناوين للبيانات وللشفرة.
/O	تحدد رقم ازعاج مختلفاً من Ox3F عند تمرير التحكم إلى الغطاء overlay.

ملحق E

المكتبات

مقدمة

هذا الملحق يناقش استخدام المكتبات مع بيسك السريع. والمكتبات هى دوال وإجراءات يكتبها المستفيدون يكون قد تم اختبارها كما أنها تكون مستقلة فى تنفيذها أى إنها لا تستخدم أى متغيرات أو موارد تكون جزءاً من برنامج آخر. ويسمح لك بيسك السريع بتحميل أحد أنواع المكتبات يسمى بالمكتبات السريعة Quick library ويكون للملف الاتساع QLB.. وعندما تحمل مثل هذه المكتبة أثناء استدعاء بيسك السريع فتصبح أجزاء المكتبة اتساعاً لمكتبة وقت تنفيذ بيسك السريع وتصبح البرامج الفرعية متاحة بدون توضيحات صريحة. والنوع الآخر للمكتبة هو نوع يستخدم فى مرحلة توصيل لترجمة ملف EXE. وهذه المكتبة لها اتساع LIB.. عندما ينتج بيسك السريع مكتبة سريعة فإنه ينتج صيغة LIB. للمكتبة لتوصيلها مع البرنامج. ويصف هذا الملحق الأنشطة التالية :

- تحميل المكتبة السريعة.
- انتاج مكتبة سريعة.
- رؤية محتويات مكتبة سريعة.
- استخدام مدير مكتبة قائم بذاته.

تحميل مكتبة سريعة

لتحميل مكتبة سريعة quick عند تشغيل بيسك السريع فإنك تبدأ عند ملقن DOS وتكتب QB/L واسم المكتبة السريعة المطلوب وتضغط على مفتاح الادخال. مثال ذلك لتحميل مكتبة سريعة اكتب QB/LQB واضغط على مفتاح الادخال. تأكد أن المكتبة السريعة المحددة تكون متاحة من أعداد مسار DOS الحالى.

انتاج مكتبة سريعة

لتعلم كيفية انتاج مكتبة سريعة من بيئة بيسك السريع حاول أن تؤدى الأنشطة التالية :

١ - نفذ بيسك السريع وانتج برنامجاً فرعياً جديداً تحت اسم InchToCm.

٢ – اكتب أسطر البرنامج التالية فى البرنامج الفرعى.

```
SUB InchToCm
   INPUT "Enter inches to convert to centimeters: "; Inches
   PRINT Inches; " inches is equal to "; (Inches * 2.54); " centimeters"
END SUB
```

٣ – أختر Make Library من قائمة Run واختر Yes فى صندوق الحوار المرحلى فاحصاً إذا ما كنت تريد حفظ الملف أم لا. اكتب Test كاسم مكتبة سريعة واختر Product Debug Code واختر Make Library.

٤ – لاحظ أسطر الأوامر التى ينتجها بيسك السريع مع انتاجه لمكتبة سريعة جديدة.

٥ – اخرج من بيسك السريع. نفذ بيسك السريع مرة أخرى وفى هذه المرة حمل المكتبة السريعة التى انتجتها.

٦ – انتقل إلى النافذة الفورية واكتب Call InchToCm ولاحظ أن البرنامج الفرعى ينفذ.

رؤية محتويات المكتبة السريعة

هناك برنامج مقدم كمثال فى قرص توزيع بيسك السريع اسمه QLBDUMP.BAS. يعطيك هذا البرنامج قائمة بمحتويات مكتبة سريعة محددة. ونستعرض فى هذا القسم محتويات المكتبة السريعة التى سبق انتاجها فى القسم السابق.

١ – حمل بيسك السريع وحمل البرنامج QLBDUMP.BAS ونفذه.

٢ – اكتب Test عند الملقن كاسم للمكتبة. مخرجات هذا البرنامج هى كما يلى :

```
Code Symbols:

  _brkctl
  _execve
  _exit
  _main
  _spawnve
  INCHTOCM

Data Symbols:

  b_erradr
  b_ULVars
  b_errlin
  b_errmod
  _end
  b_errnum
  b_ULSymSeg
  STKHQQ
  _environ
  _errno
  _edata

Press any key to continue
```

لاحظ اسم البرنامج الفرعى فى النصف الأول من المخرجات.

استخدام مدير مكتبة سريعة قائم بذاته

مدير المكتبة السريعة القائم بذاته عبارة عن برنامج مقدم على أنه ملف EXE. يسمح لك باضافة برامج فرعية أوحذفها أو نقلها أو استبدالها أو نسخها أو دمجها من مكتبة LIB. لتكوين مكتبة جديدة. واسم البرنامج هو LIB.EXE.

وفيما يلى قائمة بالعمليات المختلفة التى يمكن أن ينفذها LIB.EXE على مكتبة معينة.

الأمــر	الوصـــف
+ objfill l lib	يضيف المكتبة أو ملف Obj. للمكتبة.
- module	يحذف جزءاً معيناً من المكتبة.
- + module	يستبدل جزء محدد .
* module	ينسخ جزءاً محدداً فى ملف Obj..
- * module	ينقل جزءاً محدداً إلى ملف Obj..

يمكن استدعاء برنامج LIB بإحدى الطرق التالية :

- كتابة ما يلى عند ملقن DOS.

```
LIB oldlib /P:number Commands,listfile,newlib;
```

- حيث oldlib هو اسم المكتبة التى تغيرها أو تنتجها. (فإذا لم يوجد الاسم فيعطيك البرنامج فرصة لانتاج واحدة). و P:number/ هو حجم الصفحة للمكتبة وهو قوة 2 بين 16 و 32,768. يحدد هذا الأمر نوع العمليات التى تريد تنفيذها على المكتبة. وهى مذكورة فى الخريطة السابقة. و Listfile هو اسم ملف قائمة تقاطع الأدلة. و Newlib هو اسم المكتبة المعدلة. ويعطى اتساع BAK. لاسم المكتبة القديم إذا لم تحدد اسم مكتبة جديداً.

- اكتب LIB مع الاجابة على الأسئلة، والمدخلات تشبه جداً صيغة سطر الأوامر فيما عدى مواصفة حجم الصفحة. ولإعطاء ملفات أكثر لأى ملقن أكتب & عند نهاية السطر ويعيد البرنامج السؤال.

- قم باعداد ملف بالاستجابات المناسبة بنفس ترتيب الأسئلة مع كتابة، عند ملقن DOS، مايلى : LIB & responsefile حيث responsefile هو الملف الموجود فيه الاستجابات والذى سبق أن أعددته.

ملحق F

رسائل الخطأ

أخطاء الإستدعاء والترجمة ووقت التنفيذ

هذا الملحق يصف رسائل الخطأ التى ينتجها بيسك السريع سواء كان ذلك أثناء استدعاء بيسك السريع أو أثناء ترجمة أو تنفيذ البرنامج. والجدول التالى يسرد رموز أخطاء وقت التنفيذ والرسائل المصاحبة لها.

الرمز	الرسالة
3	العودة بدون GOSUB.
4	لا توجد بيانات.
5	استدعاء دالة غير صحيح.
6	سريان زائد.
7	الذاكرة غير كافية.
9	دليل المنظومة يقع خارج المدى.
11	قسمة على صفر.
14	مكان السلسلة لا يكفى.
16	صيغة السلسلة معقدة جداً.
19	لا يوجد RESUME.
20	يوجد RESUME بدون خطأ.
24	وقت زائد بالنسبة للوحدة.
25	خطأ فى الوحدة.
27	لا يوجد ورق.
39	يتوقع وجود CASE ELSE.
40	مطلوب متغير.
50	سريان زائد لـ FIELD.
51	خطأ داخلى.

الرمـــز	الرســـالة
52	رقم ملف أو اسم ملف خطأ.
53	الملف غير موجود.
54	حالة ملف غير صحيحة.
55	الملف مفتوح بالفعل.
56	عبارة FIELD نشطة.
57	خطأ فى مدخلات أو مخرجات الوحدة.
58	الملف موجود فعلاً.
59	طول سجل خطأ.
61	القرص مملوء.
62	تزيد المدخلات المطلوبة عن نهاية الملف.
63	رقم سجل غير صحيح.
64	اسم ملف خطأ.
67	ملفات كثيرة جداً.
68	الوحدة غير متاحة.
69	سريان زائد فى الذاكرة الاحتياطية للاتصالات.
70	تم سحب التصريح.
71	القرص غير معد.
72	خطأ فى الوسط.
73	معالم مطورة غير متاحة.
74	إعادة تسمية عبر الأقراص.
75	خطأ فى الاتصال بالملف أو المسار.
76	المسار غير موجود.

وفيما يلى قائمة برسائل الخطأ التى ينتجها بيسك السريع أثناء الاستدعاء والترجمة ويتبعها وصف موجز عندما يكون ذلك مناسباً.

Advanced features unavailable	معالم مطورة غير موجودة

(لايمكن استخدام سمات جديدة للغة مع صيغ بيسك القديمة)

Argument-count Mismatch	عداد القائمة غير متوافق
Array already dimensioned	المنظومة لها أبعاد بالفعل
Array not defined	المنظومة غير معرفة
Array not dimensioned	المنظومة لا أبعاد لها
Array too big	المنظومة كبيرة جداً

(إستخدم خيار ah/ أثناء الترجمة للمنظومات التى حجمها أكبر من 64K)

Argument-count AS clause expected	عداد القائمة متوقع
AS clause required on first declaration	جزء AS مطلوب فى أول توضيح
AS missing	AS غير موجود
Astrisk missing	النجمة غير موجودة
Bad file name	حالة ملف غير صحيحة

(استخدام عبارات مدخلات أو مخرجات غير مناسبة مع حالة ملف معينة)

Bad file mode	اسم ملف غير صحيح
Bad file name or number	رقم ملف أو اسم ملف غير صحيح
Bad record length	طول خطأ للسجل
Bad record number	رقم خطأ للسجل
BASE missing	BASE غير موجودة
Binary source file	ملف مصدر ثنائى
Block IF without END IF	مجموعة IF بدون END IF
Buffer size expected after /c:	حجم الذاكرة الاحتياطية متوقع بعد : c/

(لا يمكن ترجمة ملفات ثنائية. يجب أن تحفظ الملفات من بيسك المطور مع خيار A للتأكد من أن الملف من نوع ASCII)

BYVAL allowed only with numeric argument	مسموح بـ BYVAL مع قوائم عددية فقط
/C: buffer size too large	حجم الذاكرة الاحتياطية كبير جداً
Cannot continue	لا يمكن الاستمرار.
Cannot find file (file name). Input path:	لايمكن ايجاد مسار مدخلات الملف (اسم الملف)

(التغييرات التى حدثت فى البرنامج تمنع من استمرار التنفيذ من عند هذه النقطة).

Cannot generate listing for binary BASIC Source file	لا يمكن انتاج قائمة لملف مصدر ثنائى بالبيسك.
Cannot start with "FN"	لا يمكن أن تبدأ بـ FN.
CASE ELSE expected	CASE ELSE متوقعة.
CASE without SELECT	CASE بدون SELECT.
Choose New from Edit menu to create new SUB or function.	اختر New من قائمة Edit لانتاج برنامج فرعى جديد أو دالة جديدة.
Colon expected after /c	يتوقع وجود نقطتين رأسيتين بعد /c
Comma missing	الفاصلة غير موجودة.
COMMON and DECLARE must preced executable statements.	يجب أن تسبق COMMON و DECLARE عبارات قابلة للتنفيذ
COMMON in Quick library too small	المشاركة فى المكتبة السريعة صغيرة جداً.
COMMON name illegal	اسم المشاركة غير صحيح.
Communication-buffer overflow.	سريان زائد فى الذاكرة الاحتياطية للاتصالات.
CONST/DIM SHARED follows SUB/FUNCTION.	ثابت أو بعد مشترك يتبع دالة أو برنامج فرعى.

Control structure in IF.. THEN.. ELSE in complete.	تكوين التـحكم فى IF..THEN..ELSE غير كامل.
Data-Memory overflow.	سريان زائد لذاكرة البيانات
DECLARE required	مطلوب DECLARE
DEF FN not allowed in control statement.	غير مسموح بـ DEF FN فى عبارات تحكم.

(غير مسموح باستخدام DEF FN داخل تكوينات التحكم مثل IF..THEN.. ELSE و CASE (SELECT

DEF without END DEF	DEF بدون END DEF
DEFtype character specification illegal	مواصفة نوع DEF غير صحيحة
Device fault	خطأ فى الوحدة (خطأ فى نظم المكونات)
Device I/O error	خطأ فى مدخلات أو مخرجات الوحدة
Device timeout	وقت زائد للوحدة
Device unavailable	الوحدة غير متاحة
Disk full	القرص مملوء
Disk-media error	خطأ فى وسط القرص
Disk not ready	القرص غير معد
Division by zero	قسمة على صفر
DO without LOOP	DO بدون LOOP
Document too large	الوثيقة كبيرة جداً
Duplicate definition	تعريف مزدوج
Duplicate label	اسم مزدوج
Dynamic array elements illegal	عناصر منظومة ديناميكة غير مسموح بها (لاتستخدم VARPTR$)
Element not defined	العنصر غير معرف
ELSE without IF	ELSE بدون IF

ELSEIF without IF	ELSEIF بدون IF
END DEF without DEF	END DEF بدون IF
END IF without IF	ENF IF بدون IF
END SELECT without SELECT	END SELECT بدون SELECT
END SUB or END FUNCTION must be a last line in window	يجب أن يكون END SUB أو END FUNCTION هو آخر سطر في النافذة.
END SUB/FUNCTION without SUB/ FUNCTION	END SUB أو END FUNCTION بدون SUB أو FUNCTION
END TYPE without TYPE	END TYPE بدون TYPE
Equal sign missing	اشارة التساوى غير موجودة.
Error during QuickBASIC initial iza- tion	خطأ أثناء بدء بيسك السريع
Error in loading file (file)-Cannot find file	خطأ فى تحميل الملف - الملف غير موجود
Error in loading file (file) - Disk I/O error	خطأ فى تحميل الملف - خطأ فى مدخلات أو مخرجات القرص
Error in loading file (file (file)- DOS memory-arena error	خطأ فى تحميل الملف - خطأ فى منطقة الذاكرة لـ DOS
Error in loading file (file) - Invalid format	خطأ فى تحميل الملف - تشكيل غير صحيح.
Error in loading file (file)- Out of memory.	خطأ فى تحميل الملف - الذاكرة لا تكفى.
EXIT DO not within DO.. LOOP	جزء EXIT DO غير موجود داخل دورة DO.
EXIT not within FOR.. NEXT	جزء EXIT غير موجود داخل دورة FOR
Expected: item	عنصر متوقع. (خطأ تكوينى، نقطة البداية موجودة عند العنصر)

Expression too complex	التعبير معقد جداً
Extra file name Ignored	اسم الملف الزائد أهمل
FIELD overflow	سريان زائد فى FIELD
FIELD statement active	عبارة FIELD نشطة
	(لا يمكن استخدام GET أو PUT مع ملف وعبارة FIELD نشطة)
File already exists	الملف موجود بالفعل
File already open	الملف مفتوح فعلاً
File not found	الملف غير موجود.
File previously loaded	سبق تحميل الملف
Fixed-length string illegal	سلسلة ثابتة الطول غير صحيحة
FOR index variable already in use	متغير دليل FOR مستخدم فعلاً
FOR index variable illegal	متغير دليل FOR غير صحيح
FOR without NEXT	FOR بدون NEXT
Formal parameter specification illegal	مواصفة مؤشر رسمى غير صحيح
Formal parameters not unique	مؤشرات رسمية ليست فريدة
Function already defined	الدالة معرفة بالفعل
Function name illegal	اسم دالة غير صحيح
Function not defined	دالة غير معرفة
GOSUB missing	GOSUB غير موجودة.
GOTO missing	GOTO غير موجودة.
GOTO or GOSUB expected	يتوقع وجود GOTO أو GOSUB
Identifier cannot end with %, &, !, #, or $	لا يمكن للمعرف أن ينتهى بأى من % أو & أو ! أو # أو $
Identifier cannot include period	لا يمكن أن توجد نقطة فى المعرف
Identifier expected	يتوقع وجود معرف
Identifier too long	المعرف طويل جداً

Illegal function call	استدعاء خطأ لدالة
Illegal in direct mode	خطأ فى الحالة المباشرة (لا تدعم النافذة الحالية ذلك)
Illegal in procedure or DEF FN	خطأ فى اجراء أو فى DEF FN
Illegal number	رقم غير صحيح
Illegal outside of SUB, FUNCTION, or DEF FN	خطأ خارج أو SUB أو FUNCTION DEF FN
Illegal outside of SUB/FUNCTION	خطأ خارج SUB أو FUNCTION
Illegal outside of TYPE block	خطأ خارج مجموعة TYPE
Illegal type character in numeric constant	خطأ وجود رمز من النوع الحرفى فى ثابت عددى
$INCLUDE- file access error	خطأ فى الاتصال بملف شمول
Include file too large	ملف الشمول كبير جداً
Input file not found	ملف المدخلات غير موجود
INPUT missing	INPUT مفتقدة
Input past end file	مدخلات تزيد عن نهاية الملف
Input run-time module path: BRUN 40 is not found	مسار جزء وقت تنفيذ المدخلات (BRUN40.EXE غير موجود)
Integer between 1 and 32,767 required	مطلوب رقم صحيح يقع بين 1 و 32,767
internal error	خطأ داخلى
Internal error near XXX	خطأ داخلى بالقرب من XXX
Invalid character	رمز غير صحيح
Invalid constant	ثابت غير صحيح
Invalid DECLARE for BASIC procedure	توضيح غير صحيح لإجراء بيسك

(لا يمكن استخدام BYVAL أو ALIAS أو CDECL مع اجراءات بيسك)

Label not defined	اسم غير معرف
Label not defined: label	اسم غير معرف: إسم
Left parantheses missing	الأقواس اليسرى غير موجودة.
Line invalid. Start again	سطر غير صحيح، ابدأ مرة أخرى
Line number or label missing	رقم سطر أو اسم سطر غير موجود
Line too long	السطر طويل جداً
LOOP without DO	LOOP بدون DO
Lower bound exceeds upper bound	الحد السفلى يتعدى الحد العلوى
Math overflow	سريان زائد حسابى
$Metacommand error	خطأ شبيه الأمر
Minus sign missing	اشارة سالب غير موجودة.
Missing Event Trapping (/W) or checking Between Statements (/V) option	خيار اصطياد الأحداث (/W) أو التأكد بين العبارات (/V) مفتقد
Missing an error (/E) option	خيار عند حدوث خطأ (/E) غير موجود.
Missing resume next (/X) option	خيار افترض التالى (/X) غير موجود
Missing level code too large	رمز المستوى طويل جداً غير موجود
Module not found. Unload module from program?	الجزء غير موجود، هل يراد عدم تحميل جزء من البرنامج
Must be first statement on the line	يجب أن تكون أول عبارة على السطر
Name of subprogram illegal	اسم البرنامج الفرعى غير صحيح
Nested function definition	تعريف دالة متداخلة
NEXT missing for variable	NEXT غير موجودة لمتغير
NEXT without FOR	NEXT بدون FOR
No line number in module-name at address segment: offset	لا يوجد رقم سطر فى اسم الجزء عند عنوان القطاع : فرع

No main module. Choose Set Main Module from the Run menu to select one	لا يوجد جزء رئيسى، اختر أعداد جزء رئيسى من قائمة Run لاختيار واحد
No RESUME	لا يوجد RESUME
Not watchable	غير مرئى (المتغير الذى تم اختياره لنافذة الرؤية غير مناسب)
Numeric array illegal	منظومة عددية غير صحيحة
Only simple variables allowed	مسموح بمتغيرات بسيطة فقط
Operation requires disk	العملية تتطلب قرصاً
Option unknown: Option	الخيار غير معروف: خيار
Out of DATA	لا توجد عبارة DATA
Out of data space	لا يوجد مكان للبيانات
Out of memory	لا توجد ذاكرة كافية
Out of paper	لا يوجد ورق
Out of stack space	مكان الحزمة أصبح غير كافياً
Out of string space	مكان السلسلة أصبح غير كافياً
Overflow	سريان زائد
Overflow in numeric constant	سريان زائد فى ثابت عددى
Parameter type missmatch	نوع غير متوافق للمؤشر
Path not found	المسار غير موجود
Path/ file access error	خطأ اتصال بملف أو مسار
Permission denied	التصريح تم الغاؤه
Procedure already defined in Quick library	الإجراء معرف فعلاً فى المكتبة السريعة
Procedure too large	الإجراء كبير جداً
Program-memory overflow	سريان زائد لذاكرة البرنامج
Read error on standard input	خطأ قراءة فى مدخلات نمطية

Record/ string assignement required	مطلوب تحديد سجل أو سلسلة
Redo from start	ابدأ من البداية

(أعطيت استجابة غير صحيحة لعبارة INPUT)

Rename access disks	إعادة تسمية عبر الأقراص

(لا يمكن إعادة تسمية ملف مع تحديد مشغل جديد)

Require DOS 2.0 or later	يتطلب DOS صيغة 2.0 أو أعلى

(صيغة غير صحيحة لنظام التشغيل DOS مستخدمة للعمل فى هذه الحالة)

RESUME without error	RESUME بدون خطأ
RETURN without GOSUB	RETURN بدون GOSUB
Right parantheses missing	الأقواس اليمنى غير موجودة
SEG or BYVAL not allowed in CALLS	SEG و BYVAL غير مسموح بها فى CALLS
SELECT without END SELECT	SELECT بدون END SELECT
Semicolon missing	فاصلة منقوطة غير موجودة
Separator illegal	فاصل غير صحيح
Simple or array variable expected	يتوقع متغير بسيط أو منظومة
Skipping forward to END TYPE statement	التقدم للأمام حتى عبارة END TYPE

(خطأ فى عبارة TYPE و END TYPE يجعل بيسك السريع يهمل بقية التعريف)

Statement cannot occur within $INCLUDE file	العبارة لا يمكن أن تحدث داخل ملف شمول
Statement cannot precede SUB/ FUNCTION definition	العبارة لا يمكن أن تسبق تعريف البرنامج الفرعى أو الدالة
Statement ignored	تهمل العبارة

Statement illegal in TYPE block	عبارة غير صحيحة فى مجموعة TYPE
Statement unrecognized	عبارة غير مميزة
Statements/ labels illegal between SELECT CASE and CASE	عبارات أو أسماء غير صحيحة بين SELECT CASE و CASE.
STOP in module name at address segment/ offset	توقف فى اسم الجـــزء عند عنوان القطاع/ الفرع
String assignement required	مطلوب تحديد سلسلة
String constant required for ALIAS	مطلوب ثابت سلسلة لـ ALIAS
String expression required	مطلوب تعبير سلسلة
String formula too complex	صيغة سلسلة معقدة جداً
String space corrupt	مكان السلسلة تلف
String variable required	مطلوب متغير سلسلة
SUB or FUNCTION missing	SUB أو FUNCTION غير موجودة
SUB/ FUNCTION without END SUB/ FUNCTION	برنامج فـرعى أو دالة بدون END SUB أو END FUNCTION
Subprogram error	خطأ برنامج فرعى
Subprogram not defined	برنامج فرعى غير معرف
Subprograms not allowed in control statements	غير مسموح ببرامج فرعية فى عبارات التحكم
Subscript out of range	دليل المنظومة يقع خارج المدى المحدد له
Subscript syntax illegal	تكوين الدليل غير صحيح
Syntax error	خطأ تكوينى
Syntax error in numeric constant	خطأ تكوينى فى ثابت عددى
THEN missing	THEN غير موجودة
TO missing	TO غير موجودة
Too many arguments in function call	قيم كثيرة جداً فى نداء الدالة

Too many dimensions	أبعاد كثيرة جداً
Too many files	ملفات كثيرة جداً
Too many labels	أسماء كثيرة جداً
Too many named COMMON blocks	مجموعات مشاركة مسماه كثيرة جداً
Too many TYPE definitions	تعريفات نوع كثيرة جداً
Too many variables for INPUT	متغيرات كثيرة جداً لعبارة المدخلات
Too many variables for LINE INPUT	متغيرات كثيرة جداً لعبارة مدخلات السطر
Type missmatch	النوع غير متوافق
TYPE missing	TYPE غير موجودة
Type more than 65,535 bytes	مكتوب أكثر من 65,535 بايت
Type not defined	النوع غير معرف
TYPE statement inproperly nested	عبارة TYPE متداخلة بصورة غير مناسبة
TYPE without END TYPE	TYPE بدون END TYPE
Typed variable not allowed in expression	المتغير المكتوب غير مسموح به فى التعبير
Unexpected end of file in TYPE declaration	نهاية غير متوقعة للملف فى توضيح النوع
Unprintable error	خطأ غير قابل للطباعة
Unrecognized switch error: "QU"	خطأ مفتاح غير مميز "Qu"
Valid options: [RUN] file /AH/B/C: Buf/G/H/L [lib]/mbf/cmd string	سلسلة خيارات صحيحة
Variable-length string required	مطلوب سلسلة متغيرة الطول
Variable name not unique	اسم متغير غير فريد
Variable required	مطلوب متغير

(متغير غير موجودة فى عبارة INPUT أو LET أو READ أو SHARED أو GET و PUT)

WEND without While	WHILE بدون WEND
WHILE without WEND	WEND بدون WHILE
Wrong number of dimensions	عدد خطأ من الأبعاد

ملحق G

تمارين على بيسك السريع

١ - حول هذا الكتاب

أ - إلى أى شخص موجه هذا الكتاب؟

ب - إذا كنت تتعلم بيسك السريع فما هو التسلسل الذى يجب أن تتبعه فى دراسة هذا الكتاب؟

ج - ما هى دلالة الخطوات المرقمة فى قسم العمليات التقليدية فى كل درس؟

٢ - عرض عام لبيسك السريع

أ - ما هى السمات الخاصة ببيسك السريع - الصيغة 4.0؟

ب - ما هى التوسعات فى البيسك الموجودة فى بيسك السريع - الصيغة 4.0؟

جـ - ما هى الطرق الأخرى الممكنة لوضع بيسك السريع على القرص الصلب؟

٣ - عينة لجلسة مع بيسك السريع

أ - ماذا تكتب لبدء بيسك السريع من ملقن DOS؟

ب - كيف تكتب برنامجاً بلغة بيسك السريع؟

جـ - ما هى علامات التنقيط التى تفصل أسطر برنامج بيسك السريع؟ ومتى تكون على نفس المستوى الطبيعى؟

د - كيف تنفذ برنامجاً فى بيسك السريع؟

هـ - ما هى القوائم التى تستدعى عندما تسحب لأسفل من الاختيار؟ وكيف يمكنك أن تحفظ ملفاً على مشغل أقراص مختلف أو فى دليل مختلف؟

٤ - الترجمة من DOS

أ - تحت أى ظروف تتم ترجمة البرنامج من DOS؟

ب - ما هو الغرض من التوصيل Linking؟

جـ - ما هو نوع الملف الذى ينتج من هذا النوع من الترجمة؟

٥ - دالـة ABS

أ - ما هو نوع القيمة التى تعيدها دالة ABS؟

ب - ما هى المواقف الأخرى التى تهمل اشارة العدد؟

٦ - دالـة ASC

أ - ما هو مدى القيمة التى تعيدها دالة ASC؟

ب - ما هو متمم دالة ASC؟

جـ - ما هى الرموز غير المتوقعة؟

٧ - دالـة ATAN

أ - ما هى وحدة القيمة التى تعيدها دالة ATAN؟

ب - ما هو النوع التقليدى للقيمة التى تعيدها الدالة؟

٨ - عبارة BEEP

أ - كم عدد مرات انتاج عبارة BEEP صوتاً من الكمبيوتر؟

ب - ما هو نوع المواقف التى يمكن استخدام عبارة BEEP فيها؟

٩ - عبارتـا BLOAD و BSAVE

أ - ما هو جزء الفرع offset فى تكوين عبارة BLOAD؟

ب - ما هو جزء مدى الطول range of length فى تكوين عبارة BSAVE؟

جـ - ما هو نوع البيانات التى تتعامل معها عبارة BLOAD وعبارة BSAVE؟

د - ماهى الوحدات غير المدعمة لعبارتى BLOAD و BSAVE؟

١٠ - عبارتـا CALL و CALLS

أ - ماهو أقصى طول لاسم البرنامج الفرعى؟

ب - ما هو الفرق بين استخدام BYVAL و SEG فى عبارة CALL؟

جـ - مـا هى إجـراءات اللغـة التى يمكن اسـتدعـاؤهـا باسـتـخدام عـبـارات CALL و CALLS؟

د - متى لا تستخدم الكلمة المحجوزة CALL فى عبارة CALL؟

هـ - ما هى البرمجة للغات مختلطة؟

١١ - عبارة CALL ABSOLUTE

أ - إلى أين ينتقل التحكم فى البرنامج عندما تنفذ عبارة CALL ABSOLUTE؟

ب - ما هى المكتبة السريعة التى يجب أن تقوم بتحميلها لكى تعمل هذه العبارة؟

جـ - كيف تمرر القوائم إلى الجزء المنادى عليه؟

١٢ - عبارتـا CALL INT86OLD و CALL INTERRUPT

أ - ما هو استدعاء عمل DOS؟

ب - ما هو الازعاج interrupt؟

جـ - ما هو الفرق بين عناصر المنظومة المستخدمة فى INT86OLD و INT86XOLD؟

د - ما هى المكتبة السريعة التى يجب أن تقوم بتحميلها لكى تعمل هذه العبارات؟

هـ - فى أى الأغراض تستخدم هذه العبارات؟

١٣ - دوال CDBL و CINT و CLNG و CSNG

أ - إلى أى نوع تحول دالة CDBL القائمة؟

ب - إلى أى نوع تحول دالة CINT القائمة؟

جـ - إلى أى نوع تحول دالة CLNG القائمة؟

د - إلى أى نوع تحول دالة CSNG القائمة؟

١٤ - عبارة CHAIN

أ - هل يمكن لبرنامج (.EXE) مستقل بذاته أن يستدعى ملفات مصدر مكتوبة ببيسك السريع؟

ب - هل يستطيع برنامج ينفذ فى بيئة بيسك السريع أن يستدعى برنامجاً قائماً بذاته؟

جـ - كيف تقتسم المتغيرات مع برنامج متسلسل مع أجزاء أخرى؟

د - هل يعود تحكم البرنامج إلى برنامج متسلسل مع أجزاء أخرى؟

١٥ - دالـة CHR$

أ – ما هو نوع القيمة التى تعيدها دالة CHR$؟

ب – ما هو مدى القيم للقائمة؟

جـ – ما هى النافذة الفورية؟

١٦ - عبارة CIRCLE

أ – ما هو جزء aspect فى التكوين؟

ب – ما هى الأجزاء التى تقرر إذا ما كانت الإحداثيات مطلقة أم نسبية؟

جـ – كيف يمكنك أن ترسم قوساً مستخدماً عبارة CIRCLE؟

١٧ - عبارة CLEAR

أ – ماذا يحدث عندما تنفذ عبارة CLEAR بدون أى قائمة؟

ب – ما هى الرصة؟

جـ – كيف يمكنك أن تغير من حجم الرصة باستخدام عبارة CLEAR؟

د – متى تريد أن تغير من حجم الرصة؟

١٨ - عبارة CLOSE

أ – ماذا يحدث عندما تستخدم عبارة CLOSE بدون قائمة؟

ب – ما هى العبارات الأخرى التى تغلق الملفات؟

جـ – هل يمكن استخدام رقم الملف بعد أن يغلق الملف المناظر لهذا الرقم؟

د – هل الذاكرة الاحتياطية النهائية تكتب على القرص قبل اغلاق الملف؟

١٩ - عبارة CLS

أ – أين يكون موقع نقطة البداية بعد تنفيذ عبارة CLS؟

ب – ما هى القيم التى يمكن تمريرها إلى عبارة CLS؟

٢٠ - عبارة COLOR

أ – ما هو الغرض من عبارة COLOR؟

ب – ما هى حالات الشاشة التى تدعم حدود الشاشة؟

جـ – ما هو نوع الألوان التى لا يمكن استخدامها كخلفية؟

٢١ - دالـة COMMAND$

أ - ما هى الحالة التى تعود بها دالة COMMAND$ لمؤشر سطر الأوامر؟

ب - ما معنى القراءة من عناصر مميزة Parsing؟

جـ - متى تستخدم قوائم سطر الأوامر؟

٢٢ -عبارة COMMON

أ - ما هو الفرق بين مجموعة COMMON مسماة وأخرى غير مسماة؟

ب - متى تستخدم مجموعة COMMON المسماة؟

جـ - متى تستخدم مجموعة COMMON غير المسماة؟

د - من الذى يسبق الآخر فى مجموعة COMMON، اسماء المعرفات أو ترتيب قائمة المعرفات؟

٢٣ -الثابت CONST

أ - ما هى الأنواع المختلفة للثوابت؟

ب - ما هى أنواع البيانات المختلفة التى يمكن أن تكون ثابتاً؟

جـ - متى تستخدم الثوابت؟

د - متى تستخدم ثابتاً حرفياً بدلاً من ثابت رمزى؟

٢٤ - دالـة COS

أ - ما هو النوع التقليدى للنتيجة من دالة COS؟

ب - ما هى وحدة القيمة التى تعود من دالة COS؟

جـ - ما هو نوع دالة COS؟

٢٥ - دالـة CSRLIN

أ - ما هى المعلومات التى تعيدها دالة CSRLIN؟

ب - فى أى غرض يمكنك أن تستخدم دالة CSRLIN؟

جـ - فى البرنامج المستخدم فى قسم العملية التقليدية ما هى المؤشرات المستخدمة فى FUNCTION GCh$؟

٢٦ - دوال CVD و CVI و CVL و CVS

أ - ما هو نوع القيمة التى تعيدها الدالة CVD؟

ب - ما هو نوع القيمة التى تعيدها الدالة CVI؟

جـ - ما هو نوع القيمة التى تعيدها الدالة CVL؟

د - ما هو نوع القيمة التى تعيدها الدالة CVS؟

هـ - ما هى الدوال التى تكون عكس هذه الدوال؟

و - ما هى الطريقة الأسهل لتخزين واسترجاع بيانات باستخدام ملف على قرص؟

٢٧ - عبارة DATA

أ - ما هو عدد عناصر البيانات التى يجب أن يكون فى عبارة DATA لتجنب خطأ وقت التنفيذ؟

ب - مع أى عبارة تستخدم عبارة DATA؟

جـ - ماذا يحدث إذا ما احتوت عبارة DATA على عناصر أكثر من اللازم لعبارة READ؟

٢٨ - دالة وعبارة DATE $

أ - ما هو شكل قائمة عبارة DATE$؟

ب - فى أى شىء تستخدم دالة SATE$؟

جـ - ما هو شكل القيمة التى تعيدها دالة DATE$؟

٢٩ - عبارة DECLARE

أ - ما هو الغرض من عبارة DECLARE؟

ب - ما نوع السلاسل غير المسموح بها فى قائمة مؤشرات عبارة DECLARE؟

جـ - كيف تسرد متغيرات المنظومات فى قائمة المؤشرات؟

د - ماذا يحدث إذا لم تكن عبارة DECLARE مكتوبة من قبل المبرمج؟

هـ - ماذا يحدد جزء CDCEL؟

و - فى أى شىء يستخدم جزء ALIAS؟

ز - ما هو DGROUP؟

ح – ماذا يمكن أن يتسبب فى نقل متغير فى الذاكرة؟

ث – ما هى الطريقة الجيدة لتجميع عبارات DECLARE؟

٣٠ - عبارة DEF FN

أ – كيف تستدعى الدالة؟ متى تعرف بعبارة DEF FN؟

ب – كيف يمكنك أن تحدد نوع نتيجة الدالة؟

جـ – ما هو الفرق بين تمرير مؤشرات بواسطة دليل أو بواسطة قيمة؟

د – كيف يمكنك أن تترك دالة معرفة بواسطة DEF FN بصورة نهائية؟

٣١ - DEF SEG

أ – كيف يحسب عنوان الذاكرة؟

ب – ما هو جزء address فى تكوين DEF SEG؟

جـ – ما هو جزء مدى العنوان؟

د – ما هو القطاع التقليدى؟

٣٢ - عبارات DEFDBL و DEFINT و DEFLNG و DEFSNG و DEFSTR

أ – من أى نوع توضح عبارة DEFDBL مجموعة من الرموز؟

ب – من أى نوع توضح عبارة DEFINT مجموعة من الرموز؟

جـ – من أى نوع توضح عبارة DEFLNG مجموعة من الرموز؟

د – من أى نوع توضح عبارة DEFSNG مجموعة من الرموز؟

هـ – من أى نوع توضح عبارة DEFSTR مجموعة من الرموز؟

و – ما هى دلالة حالة الحروف فى هذه العبارات؟

ز – كيف يمكنك أن تغير نوع الحرف بعد أن تقوم بتعريفه فعلاً؟

٣٣ - عبارة DIM

أ – فى أى شيء تستخدم عبارة DIM؟

ب – ما هو أقصى عدد مسموح به لأبعاد احدى المنظومات؟

جـ - ما هو أقصى حجم لمنظومة معتادة؟

د - كيف توضح منظومة يكون حجمها أكبر من أقصى حجم؟

هـ - ما هى الأنواع المختلفة للمنظومات؟

٣٤- عبارة DO LOOP

أ - ما هى الدورة؟

ب - ما هو تكوين التحكم؟

جـ - ما هو تعبير بوليان؟

د - ما هو الاختلاف فى منطق برنامج يستخدم عبارة دورة DO تتبع التكوين الأول عن دورة DO تتبع التكوين الثانى؟

٣٥- عبارة DRAW

أ - ما هى وحدة الحركة التقليدية فى عبارة DRAW؟

ب - كيف يمكنك استخدام "TA" فى ماكرو DRAW؟

جـ - ما هو تنفيذ سلسلة جزئية؟

د - كيف يمكنك أن تغير وحدة الحركة فى عبارة DRAW؟

هـ - كيف يمكنك وصف ما إذا كانت الاحداثيات مطلقة أو نسبية؟

٣٦- عبارة END

أ - ما هى الاستخدامات المختلفة لعبارة END؟

ب - متى يمكن حذف عبارة END دون أن يحدث خطأ وقت التنفيذ؟

٣٧- عبارة ENVIRON ودالة ENVIRON$

أ - ما هى القيمة التى تعيدها دالة ENVIRON؟

ب - كيف توضح متغيرات بيئة DOS من DOS؟

جـ - كيف توضح متغيرات بيئة DOS من برنامج بيسك السريع؟

د - ماذا يحدث لاعدادات البيئة بعد انتهاء برنامج بيسك السريع؟

هـ - كيف تحصل على اعداد ملقن DOS الحالى من برنامج بيسك السريع؟

٣٨ - دالة EOF

أ - ماذا يحدث إذا حاولت أن تقرأ ملفاً بعد استنفاذ البيانات الموجودة فى الملف؟

ب - ماذا يؤدى إلى حدوث شرط نهاية الملف على وحدة اتصالات عندما تفتح الوحدة كملف ASCII؟

جـ - ما هى الوحدات التى لا يمكنك استخدامها مع دالة EOF؟

٣٩ - عبارة ERASE

أ - فى أى شىء تستخدم عبارة ERASE؟

ب - ما هو نوع المنظومات التى يمكنك استخدام عبارة ERASE معها؟

جـ - ماذا يحدث للمنظومات الديناميكية عندما تنفذ ERASE عليها؟

د - ماذا يحدث للمنظومات الاستاتيكية عندما تنفذ ERASE عليها؟

٤٠ - دالتا ERDEV و $ERDEV

أ - على أى نوع من الوحدات تستخدم دوال ERDEV و $ERDEV؟

ب - كيف يمكنك تفسير القيمة التى تعود من دالة ERDEV؟

٤١ - دالتا ERR و ERL وعبارة ON ERROR

أ - ما هو الانتخاب polling؟

ب - ما هى المعلومات التى تعيدها دالة ERR؟

جـ - ما هى المعلومات التى تعيدها دالة ERL؟

د - كيف يمكنك عمل تصيد للأحداث خاص بالأخطاء؟

٤٢ - عبارة ERROR

أ - لماذا يجب عليك أن تعرف شفرة أخطائك الخاصة بك؟

ب - ما هى الطريقة الآمنة لتحديد شفرات الأخطاء؟

جـ - كيف يمكنك استخدام عبارة ERROR فى تعريف شفرات خطأ جديدة؟

٤٣ - عبارة EXIT

أ - لماذا يجب أن تخرج من إحدى الدورات بصفة دائمة؟

ب - ما هى الأشكال المختلفة لعبارة EXIT؟

٤٤ - دالـة EXP

أ – فى أى شىء تستخدم دالة EXP؟

ب – ما هو مدى القائمة فى دالة EXP؟

٤٥ - عبارة FIELD

أ – ماذا تفعل عبارة FIELD؟

ب – ما هى الطريقة السهلة لتحقيق نفس تأثير عبارة FIELD؟

جـ – لماذا تكون عبارة FIELD أكثر نفعاً مع الملفات غير النصية؟

٤٦ - دالـة FILEATTR

أ – ما هى معالجة ملف DOS؟

ب – ما هى الخواص التى تعود من معنى دالة FILEATTR؟

جـ – أين يمكنك أن تستخدم دالة FILEATTR؟

٤٧ - عبارات FILES و CHDIR و MKDIR و RMDIR

أ – ماذا تفعل عبارة FILES؟

ب – كيف تمرر سلاسل مواصفات الملف إلى العبارات فى هذا الجزء؟

جـ – ما هى الرموز الخاصة wild-card؟

د – ما هو أقصى طول لاسم المسار فى عبارة CHDIR؟

هـ – ما هو أقصى طول لاسم المسار فى عبارة RMDIR؟

٤٨ - دالـة FIX

أ – ما هو نوع النتيجة التى تعيدها دالة FIX؟

ب – ما هى التطبيقات التى تتطلب دالة FIX؟

٤٩ - عبارة FOR.. NEXT

أ – متى يستخدم جزء STEP؟

ب – هل تستطيع أن تستخدم أرقاماً كسرية كقيم للعداد؟

جـ – ما هو التداخل؟

٥٠ - دالــة FRE

أ - ما هى المعلومات التى تقدمها دالة FRE؟

ب - كيف تجد حجم المكان المتاح للرصة؟

جـ - تحت أى ظروف تستخدم دوال FRE؟

د - كيف تجعل كل مخازن السلاسل متماسة باستخدام دالة FRE؟

٥١ - دالــة FREEFILE

أ - فى أى غرض تستخدم دالة FREEFILE؟

ب - ما هى المعلومات التى تقدمها دالة FREEFILE؟

٥٢ - عبارة FUNCTION

أ - ما هو أقصى طول مسموح به لاسم الدالة؟

ب - ماذا يفعل جزء STATIC؟

جـ - كيف تسرد مؤشرات المنظومة؟

د - كيف تترك مجموعة FUNCTION التنفيذ؟

هـ - متى تكون الدالة متسمة بسمة الاعادة الذاتية؟

٥٣ - عبارتا GET و PUT

أ - متى تستخدم عبارات GET و PUT فى تشغيل الملف؟

ب - ما هو جزء recordnum فى تكوين عبارة GET؟

جـ - لماذا يجب عدم استخدام عبارة FIELD عند استخدام عبارة GET مع جزء variable؟

د - لماذا يجب أن تتجنب استخدام عبارات GET و PUT فى الاتصالات مع سجلات ثابتة الطول؟

٥٤ - عبارتا GET graphics و PUT graphics

أ - ما هو الغرض من جزء STEP فى تكوين عبارات GET و PUT؟

ب - ما هى أهمية جزء array فى تكوين عبارة PUT؟

جـ - ما هى مؤثرات بوليان التى يمكن استخدامها مع عبارة PUT؟

٥٥ - تكوين GOSUB.. RETURN

أ - ما هو نوع البرامج الفرعية التى يمكنك استدعاؤها باستخدام عبارة GOSUB؟

ب - ما مدى العمق الذى يمكن استخدامه فى تداخل البرامج الفرعية؟

جـ - لماذا يجب أن تكتب برامج فرعية؟

د - كيف تستخدم عبارة RETURN مع اسم سطر؟

٥٦ - عبارة GOTO

أ - ما هو التفريع غير الشرطى؟

ب - متى يمكنك استخدام التفريع غير الشرطى؟

جـ - كيف يمكنك العودة إلى نفس المكان بعد تنفيذ عبارة GOTO؟

٥٧ - دالـة HEX$

أ - هل تستطيع أداء حسابات ستة عشرية على قيم تعيدها دالة HEX$؟

ب - ما هو نوع القائمة التى تقبلها دالة HEX$؟

٥٨ - عبارة IF.. THEN.. ELSE

أ - ما هو التفريع الشرطى؟

ب - ما هو الفرق بين تكوينى عبارة IF.. THEN.. ELSE؟

جـ - هل تستطيع أن تستخدم GOSUB مع THEN أثناء الاشارة إلى اسم السطر؟

٥٩ - عبارة $INCLUDE

أ - ماذا يعنى شبيه الأمر؟

ب - ما هى القيود على الملف المشمول؟

جـ - هل تستطيع تقديم اسم مسار مع اسم ملف فى عبارة $INCLUDE؟

٦٠ - دالـة INKEY$

أ - ما هى المعلومات التى تعيدها دالة INKEY$؟

ب - كيف تحذف خليط مشاوير المفاتيح باستخدام دلة INKEY$؟

جـ - ما هى مشاوير المفاتيح التى لا تستطيع اصطيادها؟

٦١ - عبارات INP و OUT و WAIT

أ - بماذا تتعامل هذه الكلمات المحجوزة؟

ب - متى تستخدم هذه الكلمات المحجوزة؟

جـ - فى عبارة WAIT فى أى ترتيب تنفذ العمليات؟

٦٢ - عبارة INPUT

أ - كيف تسرد المتغيرات فى عبارة INPUT؟

ب - كيف يمكنك أن تبطل طباعة علامة استفهام؟

جـ - كيف يجب إدخال المدخلات؟

د - أثناء الادخال ماذا يحذف محتوى المدخلات بغض النظر عن موقع نقطة البداية؟

٦٣ - عبارة #INPUT

أ - متى تستخدم عبارة #INPUT؟

ب - كيف تسرد المتغيرات فى هذه العبارة؟

جـ - متى تنتهى المدخلات من الملف؟

٦٤ - عبارة $INPUT

أ - كيف تختلف عبارة $INPUT عن عبارة #INPUT؟

ب - هل تعرض عبارة $INPUT بيانات على الشاشة؟

جـ - ما هى الوحدات التى تدعمها عبارة $INPUT؟

٦ - دالـة INSTR

أ - ما هى المعلومات التى تعيدها دالة INSTR؟

ب - كيف تختار الموقع فى السلسلة التى يجب أن يبدأ فيه البحث؟

٦٦ - دالـة INT

أ - ما هو نوع النتيجة التى تعيدها دالة INT؟

ب - هل تكون للنتيجة اشارة؟

٦٧ - دالتا IOCTL و IOCTL$

أ - متى يمكن استخدام هاتين الدالتين؟

ب - ما هى متطلبات هاتين الدالتين لتعملا؟

جـ - ما هى الوحدات غير المدعمة بواسطة هاتين الدالتين؟

٦٨ - عبارات KEY

أ - فى أى غرض تستخدم عبارات KEY؟

ب - كيف يمكنك تعريف مشوار مفتاح جديد باستخدام عبارات KEY؟

جـ - كيف يمكنك إلغاء عمل تحديد سبق حدوثه بواسطة دالة KEY؟

٦٩ - عبارة KILL

أ - هل تستطيع أن تحذف ملف يكون مفتوحاً؟

ب - متى تستخدم الرموز الخاصة wild-card فى اسم الملف؟

٧٠ - الأسماء أو LABLES

أ - ما هى الأسماء أو العناوين labels؟

ب - ما هو أقصى حجم مسموح به للاسم؟

جـ - ما هو عدد الأسماء التى يمكنك أن تكتبها على سطر واحد؟

د - متى تستخدم الأسماء؟

٧١ - دالتا LBOUND و UBOUND

أ - ما هى المعلومات التى تقدمها دالة LBOUND؟

ب - ما هى المعلومات التى تقدمها دالة UNBOUND؟

جـ - متى يمكنك استخدام هذه الدوال؟

٧٢ - دالتا LCASE$ و UCASE$

أ - ما هى الاجراءات التى تنفذها هذه الدوال على السلاسل؟

ب - متى تستخدم هذه الدوال؟

٧٣ - دالة LEFT$

أ - ماذا يحدث إذا كان جزء تعبير السلسلة فارغاً؟

ب - على أى جزء من السلسلة تعمل دالة LEFT$؟

٧٤ - دالة LEN

أ - فى أى غرض تستخدم دالة LEN؟

ب - أى تكوين تستخدمه للحصول على حجم متغير؟

٧٥ - عبارة LET

أ - ما هى الوظيفة التى تؤديها عبارة LET؟

ب - كيف يمكنك أن توضح متغيراً بدون توضيح رسمى؟

جـ - هل تستطيع أن تكتب برنامجاً بدون استخدام عبارة LET؟

٧٦ - عبارة LINE

أ - فى عبارة LINE ما هى وظيفة جزء STEP؟

ب - كيف يمكنك انتاج صندوق مملوء باستخدام عبارة LINE؟

جـ - كيف تعرف الاحداثيات المقدمة صندوقاً؟

٧٧ - عبارتا LINE INPUT و LINE INPUT#

أ - ما هو الاختلاف بين عبارتى LINE INPUT و LINE INPUT#؟

ب - متى تنتهى المدخلات؟

٧٨ - دالة LOC

أ - ما هى المعلومات التى تقدمها دالة LOC؟

ب - على أى الوحدات لا يمكنك استخدام دالة LOC؟

٧٩ - عبارة LOCATE

أ - فى أى غرض تستخدم عبارة LOCATE؟

ب - كيف يمكنك التحكم فى حجم نقطة البداية؟

جـ - كيف يمكنك التحكم فى أن تكون نقطة البداية مرئية أو غير مرئية؟

٨٠ - عبارتا LOCK و UNLOCK

أ - تحت أى ظروف يمكنك أن تستخدم عبارتى LOCK و UNLOCK؟

ب - ما هو أقصى رقم سجل؟

جـ - ما هو أقصى حجم سجل؟

٨١ - دالة EOF

أ - ما هى المعلومات التى تقدمها دالة EOF؟

ب - على أى الوحدات لا يمكنك استخدام دالة EOF؟

جـ - متى يمكنك استخدام دالة EOF؟

٨٢ - دالة LOG

أ - ما هو الأساس الذى تعيد دالة LOG نتيجتها عليه؟

ب - ما هو النوع التقليدى للنتيجة؟

٨٣ - دالة LPOS

أ - ما هى المعلومات التى تقدمها دالة LPOS؟

ب - هل القيمة التى تعيدها LPOS هى الموقع الطبيعى لرأس الطابع؟

٨٤ - عبارتا LPRINT و LPRINT USING

أ - أين تتجه المخرجات عند استخدام هذه العبارات؟

ب - ما هو العرض الذى تفترضه هذه العبارات للطابع؟

٨٥ - عبارتا LSET و RSET

أ - فى أى غرض تستخدم عبارتا LSET و RSET؟

ب - ماذا يحدث عندما يكون متغير السلسلة أكبر من البيانات؟

جـ - ما هو الاختلاف بين عبارة LSET وعبارة RSET؟

٨٦ - دالتا LTRIM$ و RTRIM$

أ - فى أى غرض تستخدم هاتان العبارتان؟

ب - ما هو الاختلاف بين دالة LTRIM$ و RTRIM$؟

٨٧ - دالة وعبارة MID$

أ - ما هو الاختلاف عندما تستخدم MID$ فى الطرف الأيسر للتحديد وعندما تستخدمها فى الطرف الأيمن من التحديد؟

ب - كيف يمكنك أن تتحكم فى الموقع الذى تستخلص منه السلسلة الجزئية؟

جـ - كيف يمكنك أن تتحكم فى الموقع الذى تضاف فيه السلسلة الجزئية؟

٨٨ - دوال MKD$ و MKI$ و MKL$ و MKS$

أ - ما هو نوع القيمة التى تعيدها دالة MKD$؟

ب - ما هو نوع القيمة التى تعيدها دالة MKI$؟

جـ - ما هو نوع القيمة التى تعيدها دالة MKL$؟

د - ما هو نوع القيمة التى تعيدها دالة MKS$؟

هـ - مع أى عبارات أخرى تستخدم هذه العبارات؟

٨٩ - دوال MKDMBF$ و MKSMBF$ و CVDMBF و CVSMBF

أ - ماذا يعنى تشكيل ميكروسوفت الثنائى؟

ب - ما هو نوع القيمة التى تعيدها MKDMBF$؟

جـ - ما هو نوع القيمة التى تعيدها MKSMBF$؟

د - ما هو نوع القيمة التى تعيدها CVDMBF؟

هـ - ما هو نوع القيمة التى تعيدها CVSMBF؟

و - ما هى مميزات تشكيل MBF على تشكيل IEEE؟

٩٠ - عبارة NAME.. AS

أ - ما هو الفرق بين أمر RENAME من DOS وعبارة NAME.. AS؟

ب - هل يمكنك إعادة تسمية دلائل باستخدام هذه العبارة؟

٩١ - دالة OCT$

أ - ما هى القيمة التى تعيدها دالة OCT$؟

ب - هل تستطيع استخدام نتيجة دالة OCT$ فى حسابات رياضية؟

٩٢ - عبارة ON event GOSUB

أ - ماذا يعنى اصطياد الأحداث؟

ب - متى تختار عملية تصيد الأحداث؟

جـ - ماذا يحدث لآلية اصطياد الأخطاء داخل جزء معالجة الأحداث؟

٩٣ - عبارتا ON.. GOTO و ON.. GOSUB

أ - ماذا يحدد كيفية التفريع إلى أرقام أسطر أو أسماء أسطر؟

ب - هل ترتيب أرقام الأسطر أو أسمائها معنوى؟

جـ - هل يمكنك استخدام خليط من أرقام أسطر وأسماء أسطر فى عبارة ON.. GO-SUB /GOTO؟

د - كيف تتشابه هذه العبارة مع عبارة IF.. THEN.. ELSE؟

٩٤ - عبارة OPEN

أ - ما هو الغرض من عبارة OPEN؟

ب - فى حالة الاتصال العشوائى RANDOM ما هو ترتيب الحالات التى تحاول عبارة OPEN أن تفتح فيها الملف قبل أن تستلم؟

جـ - ما هو القيد على حجم السجل؟

د - ما هى الوحدات المدعمة كجزء من اسم الملف؟

٩٥ - عبارتا OPEN COM و COM

أ - ما هو الغرض من عبارة OPEN COM؟

ب - ما هو الغرض من عبارة COM؟

جـ - ما هو عدد أجزاء الاتصالات المدعمة بواسطة هذه العبارات؟

د - ما هى العبارات الأخرى التى يجب أن تستخدم مع عبارة COM؟

٩٦ - عبارة OPTION BASE

أ - ما هما القيمتان المكنتان المستخدمتان مع عبارة OPTION BASE؟

ب - هل تنقل اعدادات OPTION BASE إلى برنامج متسلسل؟

جـ - كيف يمكنك أن تحدد بطريقة أخرى حدود منظومة؟

٩٧ - عبارة PRINT

أ - ما هو الاختلاف الذى يحدثه جزء STEP فى عبارة PAINT؟

ب - ماذا يعنى tiling؟

٩٨ - PALETTE و PALETTE USING

أ - ماذا تعنى مجموعة الألوان PALETTE؟

ب - ما مدى سرعة تغيير تأثير مجموعة الألوان على العرض الحالى؟

جـ - لماذا يستخدم جزء USING؟

٩٩ - عبارة PCOPY

أ - ما هو عدد صفحات الشاشة فى حالة بطاقة CGA؟

ب - متى تستخدم عبارة PCOPY؟

١٠٠ - دالة PEEK و عبارة POKE

أ - ما هو القطاع الذى يتأثر بعبارة PEEK أو عبارة POKE؟

ب - ما هو مدى جزء العنوان؟

جـ - ما هى القيمة التى تعيدها عبارة PEEK؟

د - كيف يمكنك تغيير القطاع الحالى؟

١٠١ - دالــة PEN

أ - إلى أى الوحدات فقط تستخدم دالة PEN؟

ب - ما هى العبارة التى يجب أن تنفذ قبل أن يمكن استخدام دالة PEN؟

١٠٢ - عبارات PEN ON و PEN OFF و PEN STOP

أ - على أى الوحدات تستخدم عبارات اصطياد الأحداث هذه؟

١٠٣ - دالة وعبارة PLAY

أ – ما هو الغرض من عبارة PLAY؟

ب – ما هو الغرض من دالة PLAY؟

جـ – ما هو عدد النوت الموسيقية التى يمكن لعبها فى الخلفية فى نفس الوقت؟

١٠٤ - عبارات PLAY ON و PLAY OFF و PLAY STOP

أ – ما هى الوحدات التى تتصيدها هذه العبارات؟

ب – ماذا يحدث داخل البرنامج الفرعى لتشغيل حدث PLAY؟

١٠٥ - دالـة PMAP

أ – ما هو الغرض من دالة PMAP؟

ب – لماذا تريد أن تعرف الاحداثيات المنطقية على الشاشة؟

١٠٦ - دالـة POINT

أ – كيف تستخدم دالة POINT فى العودة مرة أخرى لنقطة أثناء رسم احدى الصور؟

ب – ما هى القيمة التى تعيدها دالة POINT إذا كانت الاحداثيات خارج حدود الشاشة الحالية؟

١٠٧ - دالـة POS

أ – ما هى المعلومات التى تقدمها دالة POS؟

ب – ما هى دلالة مؤشر العمود؟

١٠٨ - عبارة PRESET

أ – ما هى دلالة جزء STEP فى التكوين؟

ب – ماذا يحدث للنقاط المرسومة عندما يحذف جزء اللون؟

١٠٩ - عبارة PRINT

أ – أين تذهب المخرجات مع عبارة PRINT؟

ب – إلى أى عدد من مناطق الطباعة تقسم الشاشة؟

جـ – ما هو عرض مناطق الطباعة بالرموز؟

د – ما هى دلالة رموز التشكيل؟

١١٠ - عبارة PRINT USING

أ – ما هى دلالة الجزء USING؟

ب – ما هو أقصى عدد أرقام يمكنك أن تحدده لبيانات عددية؟

جـ – ما هو رمز التشكيل للطباعة فى الشكل الأسى؟

١١١ - عبارتا #PRINT و #PRINT USING

أ – ما هو الاختلاف بين هاتين العبارتين وعبارتى PRINT و PRINT USING؟

١١٢ - عبارة PSET

أ – ما هو الفرق بين هذه العبارة وعبارة PRESET؟

١١٣ - عبارة RANDOMIZE

أ – لماذا تستخدم دالة RANDOMIZE؟

ب – ما هى الطريقة المقنعة لضمان أن دالة RANDOMIZE تحصل على عدد فريد؟

١١٤ - عبارة READ

أ – ما هو الغرض من عبارة READ؟

ب – ما هى العبارة الأخرى المستخدمة مع عبارة READ؟

جـ – ما هو الرمز المستخدم فى فصل المتغيرات عن بعضها البعض فى عبارة READ؟

١١٥ - عبارة REDIM

أ – ما هو دليل المنظومة؟

ب – على أى نوع من المنظومات يمكنك استخدام عبارة REDIM؟

جـ – متى تستخدم عبارة REDIM؟

١١٦ - عبارة REM

أ – ما هى الطريقة الأخرى لإدخال عبارات فى برنامج بيسك السريع؟

ب – ما هو الغرض الآخر لاستخدام عبارة REM؟

١١٧ - عبارة RESET

أ – متى تستخدم عبارة RESET؟

١١٨ - عبارة RESTORE

أ – ما هو الغرض من عبارة RESTORE؟

ب – متى تحدد رقم سطر أو اسم سطر مع عبارة RESTORE؟

١١٩ - عبارة RESUME

أ – متى تستخدم RESUME 0؟

ب – متى تستخدم RESUME NEXT؟

جـ – متى تستخدم RESUME line number line label؟

د – أين يمكن أن تحدث عبارة RESUME؟

١٢٠ - دالـة RIGHT$

أ – ماذا يحدث إذا كان جزء تعبير السلسلة فارغاً؟

ب – على أى جزء من السلسلة تعمل دالة RIGHT$؟

١٢١ - دالـة RND

أ – متى تستخدم دالة RND؟

ب – كيف تمنع دالة RND من اعادة نفس تسلسل الأعداد؟

جـ – كيف يمكنك أن تحصل على أرقام صحيحة من دالة RND؟

١٢٢ - عبارة RUN

أ – ماذا يحدث لقيم المتغيرات عندما تستخدم عبارة RUN مع مواصفة سطر؟

ب – ماذا يحدث عندما ينفذ برنامج آخر باستخدام عبارة RUN؟

١٢٣ - دالـة SADD

أ – ما هى القيمة التى تعيدها دالة SADD؟

ب – لماذا تستخدم دالة SADD؟

١٢٤ - دالة وعبارة SCREEN

أ - ما نوع مطبع العرض الذى تدعمه عبارة SCREEN؟

ب - ما هو جزء الصفحة النشطة فى عبارة SCREEN؟

جـ - ما هو جزء الصفحة المرئية فى عبارة SCREEN؟

د - ما هى المعلومات التى تقدمها دالة SCREEN؟

١٢٥ - دالة وعبارة SEEK

أ - ماذا تفعل دالة SEEK؟

ب - ماذا تفعل عبارة SEEK؟

جـ - ما هى أقصى قيمة لجزء pos فى عبارة SEEK؟

د - لماذا يجب أن تستخدم عبارة SEEK ودالة SEEK؟

١٢٦ - عبارة SELECT CASE

أ - ما هو تكوين التحكم الذى يمكن أن توفره SELECT CASE بصورة أكثر ترتيباً؟

ب - ما هو الرمز المستخدم فى فصل حالات الاختبار؟

١٢٧ - دالة SETMEM

أ - ما هو الغرض من عبارة SETMEM؟

ب - كيف يمكنك الحصول على حجم **الكومة البعيدة** مستخدماً دالة SETMEM؟

١٢٨ - دالة SGN

أ - أين يمكنك استخدام دالة SGN؟

١٢٩ - عبارة SHARED

أ - أين يمكن أن تظهر عبارة SHARED؟

ب - ما هو الغرض من عبارة SHARED؟

جـ - كيف يمكن سرد متغيرات المنظومة فى عبارة SHARED؟

١٣٠ - عبارة SHELL

أ - ما هو الأمر الذى تكتبه للعودة إلى البرنامج بعد تنفيذ عبارة SHELL بدون مؤشرات؟

ب - ما هى المؤشرات التى تقبلها عبارة SHELL؟

١٣١ - دالة SIN

أ - ما هى وحدة القيمة التى تعيدها دالة SIN؟

ب - ما هو النوع التقليدى لنتيجة دالة SIN؟

١٣٢ - عبارة SOUND

أ - ما هو مدى جزء التردد فى عبارة SOUND؟

ب - كيف يمكنك اغلاق متحدث الكمبيوتر بعد عبارة SOUND؟

١٣٣ - دالة SPACE$

أ - أين تستخدم دالة SPACE$؟

ب - ما هى الدوال التى يمكنها تحقيق نفس النتائج؟

١٣٤ - دالة SPC

أ - ما هو مدى القائمة لدالة SPC؟

ب - ما هى الدالة الأخرى التى يمكنها أن تحقق نتائج شبيهة؟

١٣٥ - دالة SQR

أ - فى أى التطبيقات يمكنك أن تستخدم دالة SQR؟

١٣٦ - عبارة STATIC

أ - ما هو الغرض من استخدام عبارة STATIC؟

ب - ماذا يحدث عندما توضح DEF FN و FUNCTION و SUB على أنها STATIC؟

١٣٧ - أشباه الأوامر $STATIC و $DYNAMIC

أ - ما هو نوع المنظومة الذى يوضحه شبه الأمر $STATIC؟

ب - ما هو نوع المنظومة الذى يوضحه شبه الأمر $DYNAMIC؟

جـ - لماذا يجب أن تستخدم أشباه الأوامر هذه؟

١٣٨ - دالة STICK

أ - عن أى وحدة تقدم دالة STICK معلومات؟

١٣٩ - عبارة STOP

أ - ما هو الغرض من عبارة STOP؟

ب - ماذا يحدث عندما تستخدم عبارة STOP فى برنامج قائم بذاته؟

جـ - ماذا يحدث عندما تستخدم عبارة STOP فى بيئة بيسك السريع؟

١٤٠ - دالة STR$

أ - لماذا تستخدم دالة STR$؟

ب - ما هو مكمل دالة STR$؟

١٤١ - دالة STRIG

أ - عن أى وحدة تقدم دالة STRIN معلومات؟

ب - ما هو مدى جزء n فى التكوين؟

١٤٢ - عبارات STRING ON و STRING OFF و STRING STOP

أ - أى الأحداث تتصيدها هذه العبارات؟

ب - ما هو مدى جزء n فى التكوين؟

١٤٣ - دالة STRING$

أ - ما هى التطبيقات الممكنة لدالة STRING$؟

ب - ما هو أكبر حجم للسلسلة يمكنك أن تحدده فى دالة STRING$؟

١٤٤ - عبارتا SUB و END SUB

أ - هل يستطيع SUB أن يستدعى نفسه؟

ب - لماذا تكتب برامج فرعية؟

جـ - كيف يمكنك إنتاج برنامج فرعى فى بيئة بيسك السريع؟

١٤٥ - عبارة SWAP

أ - ماذا تفعل عبارة SWAP؟

ب - ما هى التطبيقات الممكنة لعبارة SWAP؟

جـ - هل يمكن أن تكون قائمة العبارة من أى نوع؟

١٤٦ - عبارة SYSTEM

أ - ماذا تفعل عبارة SYSTEM؟

ب - هل يوجد لعبارة SYSTEM أى مؤشرات؟

جـ - ما هو الاختلاف بين تنفيذ برنامج قائم بذاته لعبارة SYSTEM وما إذا كان البرنامج ينفذ عندما ينفذ بيسك السريع عبارة SYSTEM؟

١٤٧ - دالة TAB

أ - ما هو الغرض من دالة TAB؟

ب - ماذا يحدث إذا كانت قائمة الدالة أكبر من عدد الأعمدة المتاحة؟

١٤٨ - دالة TAN

أ - ما هى نتيجة دالة TAN؟

ب - ما هو النوع التقليدى لنتيجة دالة TAN؟

١٤٩ - دالة وعبارة TIME$

أ - ما هو شكل وضع الوقت باستخدام عبارة TIME$؟

ب - ما هو شكل وضع الوقت الذى تعيده دالة STIME$؟

١٥٠ - دالة TIMER

أ - ما هى المعلومات التى تقدمها دالة TIMER؟

ب - ما هى التطبيقات التى تناسبها دالة TIMER؟

١٥١ - عبارات TIMER ON و TIMER OFF و TIMER STOP

أ - أى الأحداث تتصيدها هذه العبارات؟

ب - ما هى العبارة الأخرى التى يجب أن تستخدم مع هذه العبارات؟

١٥٢ - عبارتا TRON و TROFF

أ - ما هو التتبع tracing؟

ب - كيف يمكن استخدام عبارتى TRON و TROFF فى البرنامج؟

جـ - كيف يمكنك تتبع البرامج بطريقة أخرى؟

١٥٣ - عبارتا TYPE و END..TYPE

أ - ما هو الغرض مَن عبارتى TYPE و END..TYPE ؟

ب - ما هى العبارة الأخرى التى يجب استخدامها قبل أن يصبح نوع البيانات متاحاً فى البرنامج؟

١٥٤ - دالة VAL

أ - ما هى الوظيفة التى تكملها دالة VAL؟

ب - عند استخدام دالة VAL فأين يتوقف التحويل؟

١٥٥ - المتغيرات

أ - ما هو المتغير؟

ب - كيف توضح أحد المتغيرات فى بيسك السريع؟

جـ - ما هى خصائص المتغيرات فى بيسك السريع؟

١٥٦ - دالتا VARPTR و VARSEG

أ - ما هو الفرع؟

ب - ما هو القطاع؟

جـ - أين يمكنك استخدام هاتين الدالتين؟

١٥٧ - دالة VARPTR$

أ - ما النتيجة التى تخرج من دالة VARPTR$؟

ب - أين يمكنك استخدام دالة VARPTR$؟

جـ - ما هو نوع المنظومات المستخدمة كمؤشرات لهذه الدالة؟

١٥٨ - عبارة VIEW

أ - ماذا تفعل عبارة VIEW؟

ب - ما هو الغرض من جزء SCREEN فى التكوين؟

جـ - ما هو مدى الرؤية view portal؟

١٥٩ - عبارة VIEW PRINT

أ - ما هو الاختلاف بين عبارة VIEW وعبارة VIEW PRINT؟

ب - كيف تؤثر العبارة على العرض المرئى؟

١٦٠ - عبارة WHILE.. WEND

أ - ما هو نوع عبارة WHILE.. WEND؟

ب - ما هى العبارات الأخرى التى تؤدى وظائف مماثلة؟

١٦١ - عبارة WIDTH

أ - أى وحدة تؤثر عليها عبارة WIDTH؟

ب - كيف تؤثر عبارة WIDTH بطرق أخرى على المخرجات؟

جـ - ما مدى سرعة اتخاذ عبارة WIDTH إجراءً ؟

١٦٢ - عبارة WINDOW

أ - ماذا تفعل عبارة WINDOW؟

ب - ما هى معنوية جزء SCREEN فى التكوين؟

١٦٣ - عبارتا WRITE و #WRITE

أ - ما هو الاختلاف بين عبارة WRITE وعبارة #WRITE؟

ب - ما هو الاختلاف بين عبارتى WRITE و #WRITE وعبارتى PRINT و #PRINT؟

قائمة بأهم المصطلحات

$DYNAMIC,	شبيه أمر ديناميكى
$INCLUDE, Metacommand,	شبيه أمر الشمول
$STATIC,	شبيه أمر أستاتيكى
ABS,	دالة القيمة المطلقة
Array, Maximum size for an,	أقصى حد للمنظومة
Array for graphics GET and PUT, Calculating size of,	حساب حجم منظومة لرسومات GET و PUT
Array subscript, Lower limit of,	الحد السفلى لدليل المنظومة
Array subscript, Upper limit of,	الحد العلوى لدليل المنظومة
Arrays	منظومات
Changing the size of, 274	تغير الحجم
Declaring,	توضيح
Dynamic,	ديناميكية
Erasing,	حذف
Initializing,	وضع قيم ابتدائية
ASC,	دالة اسكى
Aspect,	وجه
ATN,	دالة قوس الظل
BASIC language implementation,	تنفيذ لغة البيسك
BEEP,	عبارة الصفير
Bits per pixel plane,	بت فى كل سطح نقاط رسم
BLOAD,	عبارة تحميل صورة الذاكرة
Branching, Conditional,	تفريع شرطى
Branching, Unconditional,	تفريع غير شرطى

BSAVE,	عبارة نسخ بيانات من الذاكرة
Buffer flushing,	تدفق الذاكرة الأحتياطية
BYVAL	كلمة تعرف كيف تمرر القائمة إلى البرنامج الفرعى
CALL,	عبارة استدعاء
CALL ABSOLUTE,	عبارة استدعاء مطلق
CALL INT86OLD,	عبارة تؤدى استدعاء DOS
CALL INT86OLD,	عبارة تؤدى استدعاء DOS
CALL INTERRUPT,	عبارة تؤدى استدعاء DOS
CALL INTERRUPTX,	عبارة تؤدى استدعاء DOS
CALLS,	عبارة استدعاء
Calling Non-Quick BASIC procedures,	استدعاء اجراءات غير بيسك السريع
Calling Quick BASIC procedures,	استدعاء اجراءات بيسك سريع
CDBL,	دالة لتحويل عددى إلى قيمة مذدوجة الدقة
CHAIN,	عبارة السلسلة
Characters, Non-printable,	رموز لاتطبع
CHDIR,	عبارة تغيير الدليل
CHR$,	دالة الرمز
CINT,	دالة لتحويل تعبير عددى إلى قيمة صحيحة
CIRCLE,	عبارة الدائرة
CLEAR,	عبارة الاخلاء
CLNG,	دالة لتحويل تعبير عددى إلى قيمة صحيحة طويلة
CLOSE,	عبارة اغلق
CLS,	عبارة اخلاء الشاشة
COLOR,	عبارة اللون

Finding text,	إيجاد نص
FIX,	دالة اعادة الرقم الصحيح
Flow control, Program,	التحكم فى مسار البرنامج
FOR,	عباره من
FRE,	دالة لتحديد الذاكرة المتاحة
FREEFILE,	دالة لتقديم رقم الملف المثالى المتاح
FUNCTION,	عباره لتعريف دالة بيسك السريع
Function, Recursive,	دالة اعادة ذاتية
GET,	عباره لقراء بيانات من ملف
GET, Graphics,	عباره للتعامل مع الرسومات
GOSUB,	تكوين لاستدعاء مقطع فرعى
GOSUB, Nesting,	تكوين لتداخل استدعاء مقاطع فرعية
GOTO,	عبارة اذهب إلى
Help, Context sensitive,	مساعدة حساسه المحتوى
HEX$,	دالة السادس عشرى
IF,	عبارة اذا
Immediate window,	نافذه فورية
Including files,	ملفات شمول
INKEY$,	دالة لاعادة سلسلة من بث أو اثنين
INP,	عبارة لقراء بايت من بوابه الآلة
INPUT#,	عباره مدخلات من ملف
INPUT$,	عباره مدخلات سلسلة من ملف
INPUT,	عباره مدخلات من لوحة مفاتيح
Input, Data,	بيانات مدخلات
Input from a file, Data,	بيانات مدخلات من ملف
INSTR,	دالة تتعامل مع السلاسل
INT,	داله لاعاده القيمه الصحيحة

IOCTL$,	داله للتداخل مع مشغل الوحده
IOCTL,	داله للتداخل مع مشغل الوحده
Joystick control,	التحكم فى عصا اللعب
KEY,	عباره لتحديد قيمة لمفاتيح الوظائف
KEYLIST,	عباره لسرد تحديدات السلاسل لمفاتيح الوظائف على الشاشة
KEY(N) OFF,	عبارة لإلغاء اصطياد الحدث للمفتاح N لحظيا
KEY (N), ON,	عبارة لتنشيط اصطياد الحدث للمفتاح N
KEY(N) STOP,	عبارة لإيقاف اصطياد الحدث للمفتاح N
Key sequences, User-defined,	تتابع مفاتيح يحدده المستفيد
Keyboard flags,	أشارات لوحة المفاتيح
Keyboard scan codes,	شفرات فحص لوحة المفاتيح
Keys, Input editing,	مفاتيح تنقيح المدخلات
Keys, Trapping extended,	مفاتيح توسيع الاصطياد
KILL,	عبارة لإلغاء احد الملفات
Lables in QuickBASIC,	الاسماء فى بيسك السريع
LBQUND,	دالة الحد السفلى لدليل بعد المنظومة
LCASE$,	دالة تحول قيمة سلسلة الى الحالة السفلية
LEPT$	دالة لإعادة رموز من ناحية اليسار
LEN,	دالة لإعطاء طول السلسلة
LET,	دالة دع
Libraries, QuickBASIC,	مكتبات بيسك السريع
Lightpen control,	التحكم فى القلم الضوئى
LINE,	عباره لرسم خط
LINE INPUT#,	عبارة لقبول مدخلات خط من ملف تتابعى
LINE INPUT,	عبارة لقبول مدخلات خط من لوحة المفاتيح
Line styling,	شكل الخط

Link options,	بدائل التوصيل
Linking,	توصيل
Loading a program,	تحميل برنامج
LOC,	دالة لتحديد موقع فى ملف
LOCATE,	عبارة لتحديد موقع نقطه البداية على الشاشة
LOCK,	عبارة اغلاق
LOF,	دالة لتقديم حجم الملف
LOG,	دالة اللوغاريتم الطبيعى
LOOP,	عبارة الدورة
LPOS,	دالة لتحديد موقع رأس الطابع
LPRINT,	دالة للطباعة بالطابع
LSET,	عبارة لنقل بيانات من الذاكرة إلى ذاكرة الملف الأحتياطية
LTRIM$,	دالة لإزالة الفراغات
Memory	ذاكرة
Available,	متاحة
Loading muliple modules in,	تحميل مقاطع متعددة فى
Reading from,	قراءة من
Writing to,	كتابة فى
Memory address,	عنوان ذاكرة
Memory organization,	تنظيم الذاكرة
Metacommands,	اشباه اوامر
Metacommands, Writing,	كتابة أشباه الأوامر
Microsoft Binary Format,	شكل ثنائى من ميكروسوفت
MID$,	دالة أو عبارة مع سلسلة مؤشر
MKD$,	دالة تحويل تعبير مزدوج الدقة إلى قيمة سلسلة من 8 بايت

MKDIR,	عبارة لعمل دليل
MKDMBF$,	دالة للتعامل مع أشكال ثنائية من ميكروسوفت
MKI$,	دالة لتحويل تعبير صحيح إلى قيمة سلسلة من 2 بايت
MKL$,	دالة لتحويل تعبير صحيح طويل إلى قيمة سلسلة من 4 بايت
MKS$,	دالة لتحويل تعبير فردى الدقة إلى قيمة سلسلة من 4 بايت
MKSMBF$,	دالة للتعامل مع أشكال ثنائية من مايكروسوفت
Music, Playing,	لعب موسيقى
Music macro,	ماكرو موسيقى
NAME...AS,	دالة لإعادة تسمية الملفات
Nesting,	تتداخل
NEXT,	عبارة التالى
OCTS,	دالة لإعادة المكافئ الثنائى لتعبير ثمانى
ON,	عبارة عند
ON ERROR GOTO,	عبارة عند حدوث خطأ إذهب إلى
OPEN,	عبارة أفتح
OPTION BASE,	عبارة أساس البديل
OUT,	عبارة لأرسال بايت إلى بوابة الآله
Output	مخرجات
Controlling,	التحكم فى
Formatting,	تشكيل
Writing,	كتابة
PAINT,	عبارة الدهان
PALETTE,	عبارة تغيير الألوان
Parameter passing,	مرور المؤشر

Parse,	قراءة من عناصر مميزة
Pasting text,	لصق النص
PCOPY,	عبارة نسخ الشاشة
PEEK,	دالة لتحديد موقع ذاكرة
PEN,	دالة إحداثيات القلم الضوئى
PEN OFF,	دالة لإلغاء اصطياد نشاط القلم الضوئى
PEN ON,	دالة لتنشيط اصطياد نشاط القلم الضوئى
PEN STOP,	دالة لإيقاف اصطياد نشاط القلم الضوئى
Pi, Value of,	قيمة ط (Π)
Planes,	مستويات
PLAY,	دالة وعبارة لعب الموسيقى
PLAY OFF,	عبارة لإلغاء اصطياد الحدث
PLAY ON,	عبارة لتنشيط اصطياد الحدث
PLAY STOP,	عبارة لإيقاف اصطياد الحدث
Plotting points,	نقاط رسم
PMAP,	دالة للتعامل مع الأحداثيات
POINT,	دالة لأحضار رقم أو احداثيات نقاط الرسم
POKE,	دالة لكتابة بيانات فى موقع ذاكرة
POS,	دالة تحدد عمود نقطة البداية
PRINT#,	عبارة كتابة فى ملف
PRINT,	عبارة كتابة على الشاشة
PRINT, USING,	عبارة الكتابة بأستخدام شكل معين
Printing data,	طباعة بيانات
PSET,	عبارة لرسم نقطة على الشاشة
PUT,	عبارة لكتابة بيانات فى ملف
PUT, Graphics,	عبارة للتعامل مع رسومات
QuickBASIC	بيسك السريع

Exiting,	خروج
Installing,	تشييد
Running,	تشغيل
QuickBASIC disks, contents,	محتويات أقراص بيسك السريع
QuickBASIC environment,	بيئة بيسك السريع
Random number generator, Reseeding,	إعادة وضع قيمة للاساس لمنتج أرقام عشوائية
Random numbers, Generating,	إنتاج أرقام عشوائية
RANDOMIZE,	عبارة لبدأ أنتاج أرقام عشوائية
READ,	عبارة قراءة
REDIM,	عبارة للتعامل مع منظومات
REM,	عبارة ملحوظات
Replacing text,	إحلال نص
RESET,	عبارة لإغلاق كل ملفات القرص
RESTORE,	عبارة لإستعادة برامج بيسك
RESUME,	عبارة لاستمرار تنفيذ البرنامج
RETURN,	عبارة للعودة من ملف فرعي
RIGHT$,	دالة لإعادة عدد رموز من ناحية اليمين
RMDIR,	عبارة لإزالة دليل
RND,	دالة تعيد رقم عشوائى
RSET,	عبارة لنقل بيانات من الذاكرة إلى ذاكرة الملف الأحتياطية
RTRIM$,	دالة لإزالة الفراغات
RUN,	عبارة تشغيل
Running LIB.EXE,	تشغيل ملف LIB. EXE
SADD,	دالة لتقديم عنوان تعبير سلسلة
Saving a program,	حفظ البرنامج

SCREEN,	دالة وعبارة الشاشة
Screen coordinates,	إحداثيات الشاشة
Screen coordinates, Relative,	إحداثيات الشاشة النسبية
Screen modes,	حالات الشاشة
Screen pages,	صفحات الشاشة
SEEK,	دالة وعبارة البحث
SEG,	كلمة تعرف كيف تمرر القائمة إلى البرنامج الفرعى
SELECT CASE,	عبارة أختيار الحالة
SETMEM,	عبارة للتعامل مع الكومة البعيدة
SGN,	دالة الإشارة
SHARED,	عبارة مشاركة
SHELL,	عبارة خروج من البرنامج وتنفيذ أمر DOS والعودة
SIN,	دالة الجيب
SOUND,	دالة الصوت
SPACE$,	دالة الفراغات
SPC,	دالة القفز
SQR,	دالة الجذر التربيعى
Stack, Modifying the size of,	تعديل حجم الرصة
STATIC,	عبارة للتعامل مع المتغيرات والمنظومات
STICK,	دالة لقراءة إحداثيات عصا اللعب
STOP,	دالة لإيقاف تنفيذ البرنامج
STR$,	دالة تحويل تعبير عددى إلى سلسلة
STRIG,	دالة لتعيد حالة إطلاق عصا اللعب
STRIG OFF,	عبارة لإلغاء تنشيط اصطياد حدث على عصا اللعب

STRIG ON,	عبارة تنشيط اصطياد حدث على عصا اللعب
STRIG STOP,	عبارة إيقاف تنشيط اصطياد حدث على عصا اللعب
STRING$,	دالة تعيد سلسلة لأحد رموز اسكى
String, Case conversion of,	تغيير حالة السلسلة
String storage compaction,	ضغط تخزين السلسلة
Stringd, Manipulating,	معالجة السلاسل
Strings, Parsing,	قراءة من سلاسل مميزة
SUB..END SUB,	عبارتان تعرفان برنامج فرعى
Subprograms,	برامج فرعية
Substrings exceution,	تنفيذ سلسلة فرعية
SWAP,	عبارة لتبادل متغيرات مؤشر
SYSTEM,	عبارة لاغلاق كل الملفات والعودة إلى نظام التشغيل
TAB,	دالة لترحيل مخرجات البرنامج للداخل
TAN,	دالة الظل
Terminating Program,	برنامج أنهاء
THEN,	بعد ذلك
Tiling,	طلاء
TIME$,	دالة وعبارة الوقت
Time, Reading the system,	قراءة وقت النظام
Time, Setting the system,	وضع وقت النظام
TIMER,	دالة تعطى الوقت المنقضى بعد الساعة ١٢ بعد الظهر
TIMER OFF,	عبارة لإلغاء تنشيط اصطياد الحدث لدالة TIMER
TIMER ON,	عبارة تنشيط اصطياد الحدث لدالة TIMER

VARPTR,	دالة للحصول على قطاع وفرع الذاكرة لمتغير
VARSEG,	دالة للحصول على قطاع وفرع الذاكرة لمتغير
VIEW,	عبارة تعرف حدود الرسومات للشاشة
VIEW PRINT,	عبارة تعرف حدود النصوص للشاشة
WAIT,	عبارة إنتظار
WHILE.. WEND,	عبارة لتكرار تنفيذ مجموعة عبارات
WIDTH,	عبارة عرض السطر
WIDTH LPRINT,	عبارة عرض سطر الطابع
WINDOW,	عبارة النافذة
WRITE#,	عبارة الكتابة فى ملف
WRITE,	عبارة الكتابة على الشاشة
Writing subroutines,	كتابة مقاطع فرعية

رقــم الإيــداع

٩٣ / ٨٦٩٣


مطابع المكتب المصرى الحديث
MODERN EGYPTIAN PRESS
ت : ٢٢١١٠٧١ - ٢٢١١٠٧٢ - فاكس ٢٢١١٠٧٣

هذا الكتاب

لقد ظهرت لغة البيسك منذ خمسة وعشرين عاماً كلغة بسيطة لتعليم المبتدئين أساسيات البرمجة باستخدام الكمبيوتر. وشهدت هذه اللغة البسيطة تطورات هائلة على مدار هذه السنين جعلتها فى مصاف لغات البرمجة الأساسية حالياً. ويرجع الفضل فى جعل هذه اللغة قادرة على دعم البرمجة المرتبة إلى صيغ كويك بيسك أو بيسك السريع.

وفيما يلى بعض مميزات صيغ بيسك السريع عن صيغ بيسك السابقة:

- إمكانية كتابة الأسطر دون وضع أرقام لها.
- استخدام نوع جديد من أنواع المتغيرات العددية وهو الرقم الصحيح الطويل long integer.
- التوسع فى مكونات التحكم التى سبق استخدامها فى صيغ البيسك السابقة لها بادخال مجموعات IF ومجموعات SELECT CASE ودورات DO
- استخدام البرامج الفرعية والتى تدعم البرمجة المرتبة.
- استخدام الملفات الثنائية، واستخدام مترجمات compilers للغة لترجمة البرامج.
- استخدام المقاطع التى يمكن معالجتها كأجزاء مستقلة، واستخدام خاصية مشاركة المتغيرات عبر المقاطع المختلفة.

وترجع أهمية هذا الكتاب لحداثة المادة العلمية المقدمة فيه وكذلك لسهولة عرضها وترتيب مواضيعها: وعلى الرغم من أن الكتاب مقدم للقارئ المبتدئ وكذلك للمبرمجين إلا أنه لا يحتاج لشىء فى دراسته سوى الالمام باستخدام لوحة مفاتيح الكمبيوتر ونظام تشغيله. وهذه متطلبات ميسورة لمن يرغب فى تعلم هذه اللغة المهمة.

والله الموفق

الناشر

عادل حسن قنديل

ISBN : 977- 5201- 46- 2

ACADEMIC BOOKSHOP

